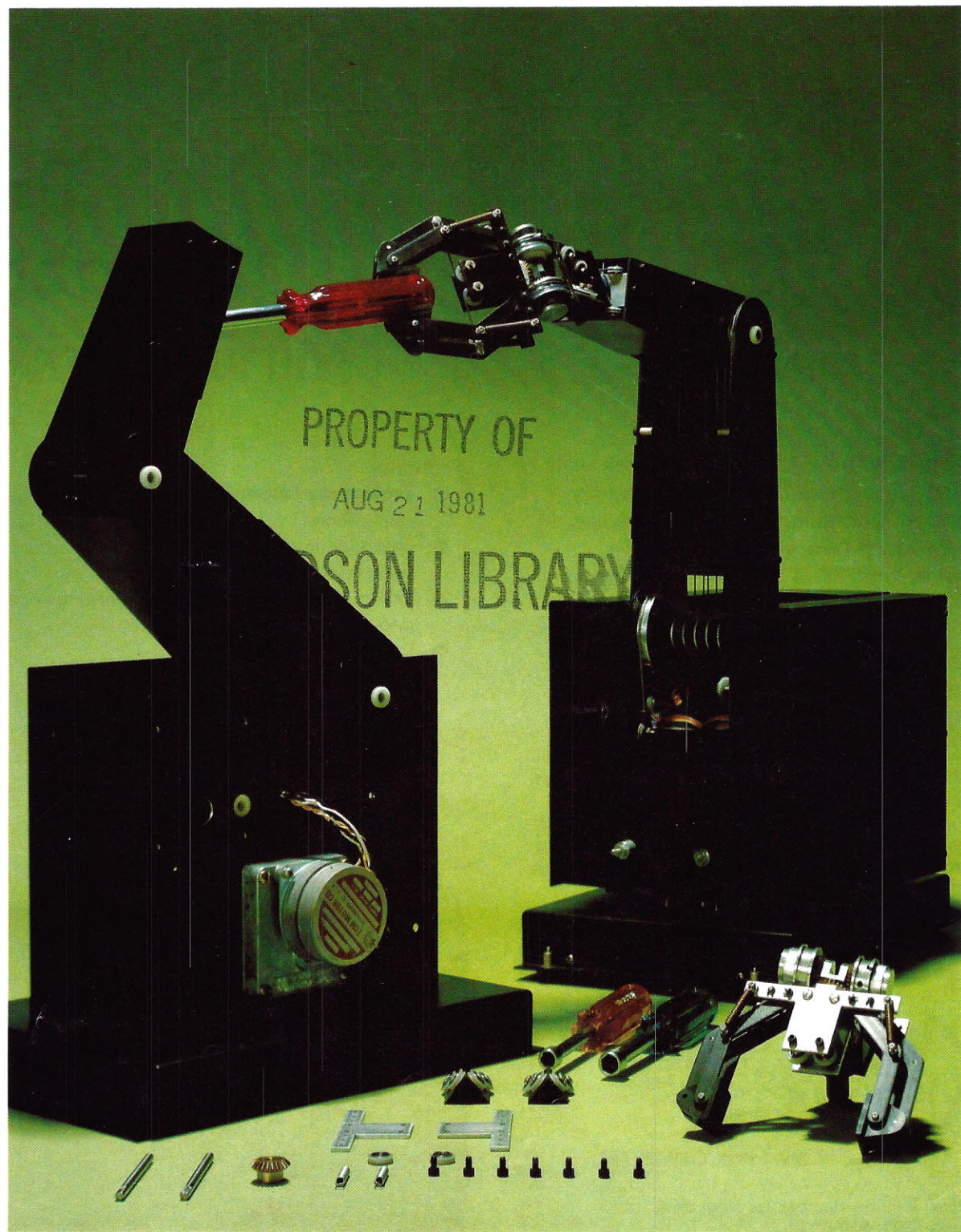
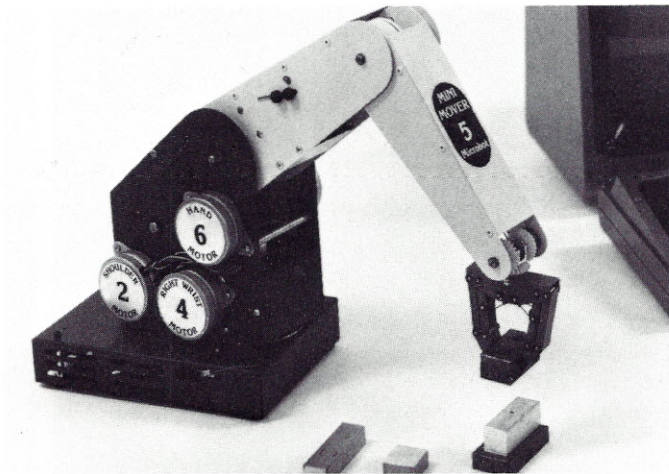


ROBOTICS AGETM

The Journal of Intelligent MachinesTM • JULY/AUG 1981 • \$3.00 USA/\$3.50 CANADA



Low Cost Table Top Robot Arm



The Microbot MiniMover-5

The MiniMover-5 is a five jointed arm with a lifting capacity of 8 ounces when fully extended. Controlled by stepper motors, it has .013 inch (worst-case) resolution, allowing **PRECISE CONTROL**. The gripper end point may be positioned in the front half of a sphere with a radius of 17.5 inches. Speed ranges from 2 to 6 inches per second, depending upon the weight of the handled object.

- **CABLE CONTROLLED** from stepper motors in the body. This allows the MiniMover-5 to move faster and provide greater lifting capacity.
- **UNIQUE DESIGN** combining features from industrial robots and remote manipulators used in severe environments. This permits a lower cost by an order of magnitude.
- **QUALITY CONSTRUCTION** ensures long life and easy maintenance.



P.O. Box 17329, Irvine, Calif. 92713

New Toll Free Number, for order desk
only (800) 854-8230.
All other calls (714) 558-8813

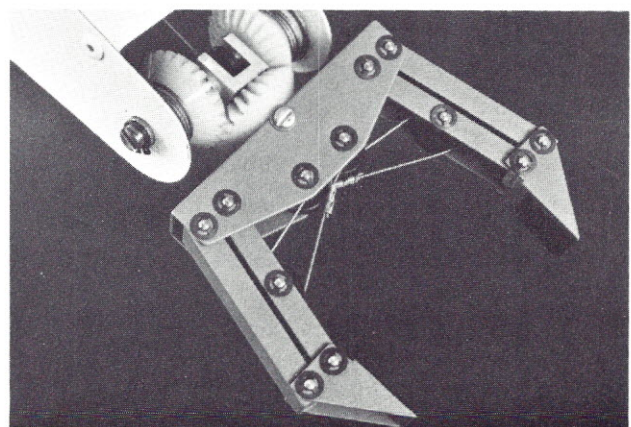
Perfect for:

- Evaluation of robots for industrial applications.
- Education in robotics principals and control.
- Experiments with artificial intelligence and robotics.
- Computer art and games.

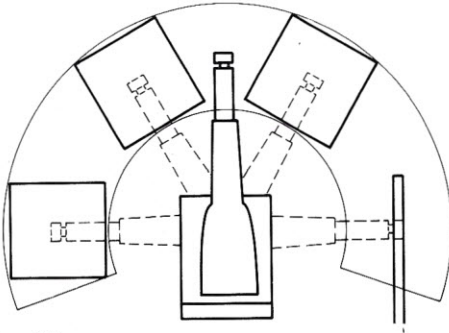
Assembled as above

ONLY \$1695

100 Page Reference and
Application Manual \$16.95



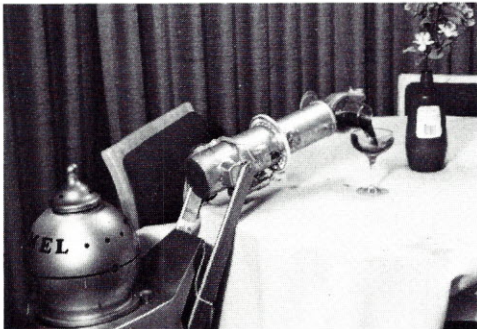
Articles



Page 26

- 4 Segmenting Binary Images** *by Robert Cunningham*
How to find regions in an image and describe their shapes.

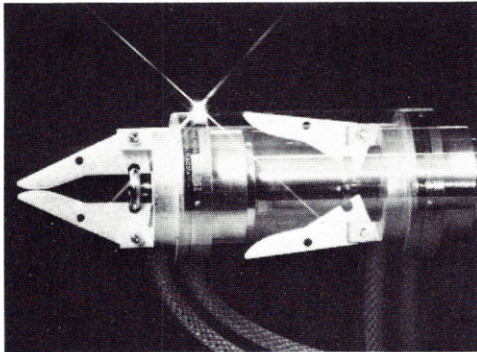
- 16 The Robot as Transfer Device** *by Vernon Mangold*
Throw away your forklift. Robots may be the best way to move objects around the factory.



Page 41

- 20 Continuous Path Control of Stepper Motors** *by Shafi Motiwalla*
With a little bit of knowledge and foresight, you can accurately track curves at high speeds—even without feedback.

- 38 TIMEL: A Homebuilt Robot** *by John Blankenship*
By ingeniously using commonplace objects, you can build an inexpensive robot for intelligent software development.



Page 45

Departments

- | | |
|---------------------------|-------------------------|
| 42 Eye to Industry | 50 Media Sensors |
| 44 New Products | 52 Organizations |
| 49 Books | |

Cover: MAX—by Manhattan Engineering, Manhattan Beach, CA.
Photo: John R. Jones

Publisher

Philip C. Flora

Managing Editor

Joel M. Wilf

Art Director

Robert Tinney

Technical Art

Heather Marr
James Marr

Advertising

Robotics Publishing Corp.
10049 Commerce Ave.
Tujunga, Calif. 91042
(213)352-7937

Editorial Director

Alan M. Thompson

Industrial Editor

J. W. Saveriano

Production Editors

Cindee Huff
Adrienne Wills-DeVine

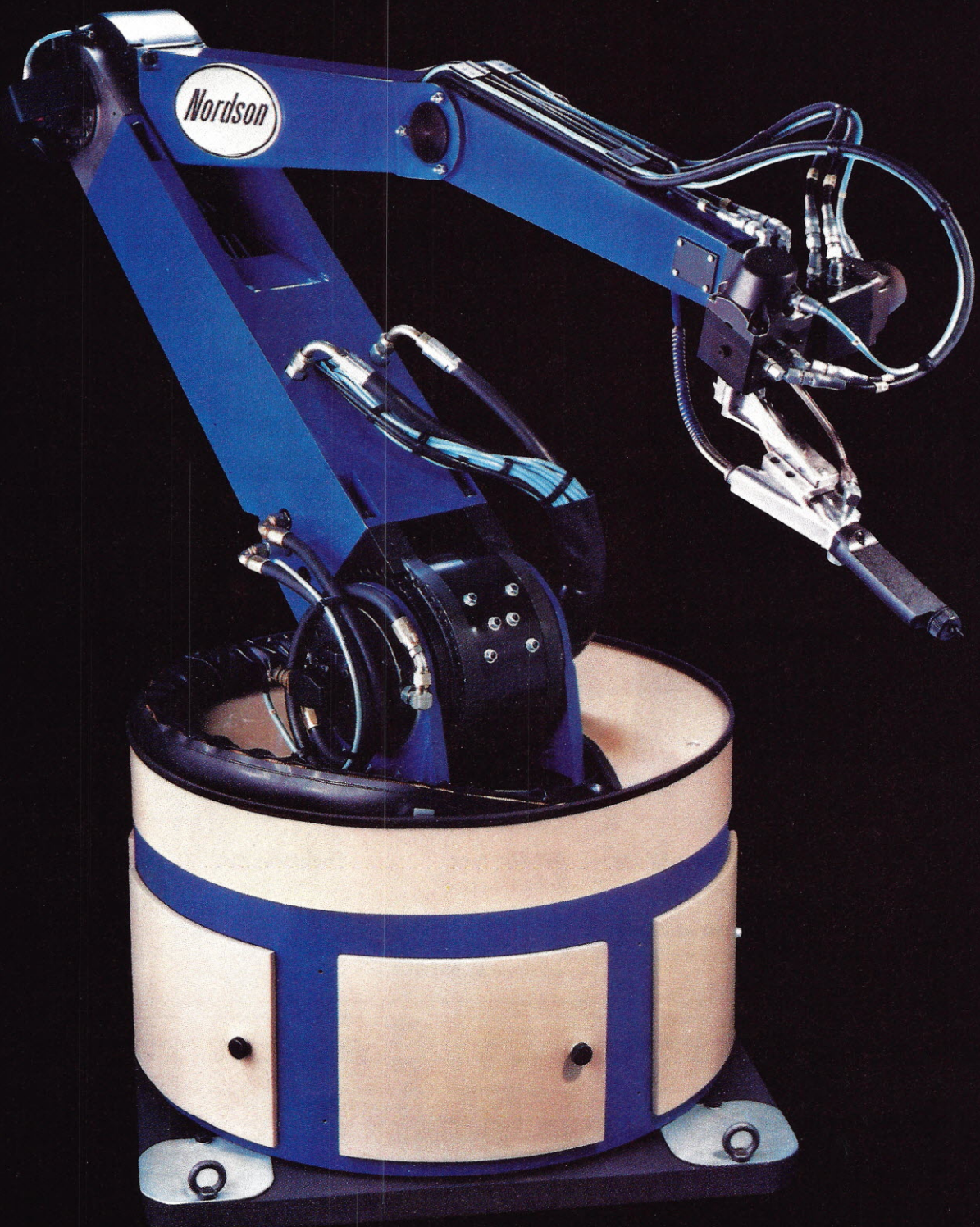
Graphics

Steven Fagerquist

ROBOTICS AGE—USPS 523850 (ISSN 0197-1905) is published six times a year, in January, March, May, July, September, and November, by Robotics Publishing Corporation, 10049 Commerce Ave., Tujunga, California 91042. Subscriptions are \$15.00 for one year in the USA. Second-class Postage paid at Tujunga, CA 91042. POSTMASTER: Send address changes to P. O. Box 423, Tujunga, CA 91042.

Subscriptions from Canada from Mexico are \$17 for one year. Other foreign countries \$19 for one year surface delivery. Single copy price is \$3.00 in the U.S. and \$3.50 in Canada and Mexico, \$4 in Europe, and \$4.50 elsewhere. Foreign subscriptions and sales should be remitted in United States funds drawn on a US bank. Address all correspondence to: Robotics Age, P. O. Box 423, Tujunga, CA 91042, phone 213/352-7937. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of ROBOTICS AGE. Each separate contribution to this issue, and the issue as a collective work ©1980 by Robotics Publishing Corporation. All rights reserved.

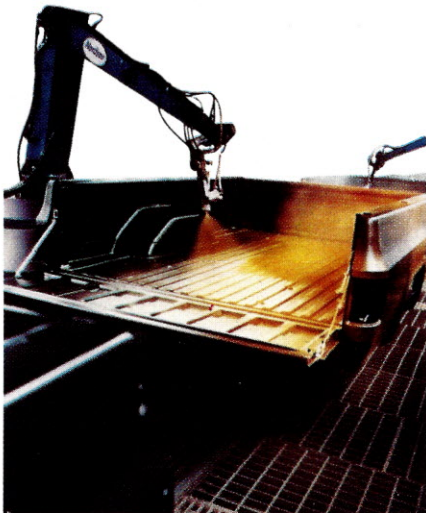
More and more coats tailored by



products wear Nordson® robots.

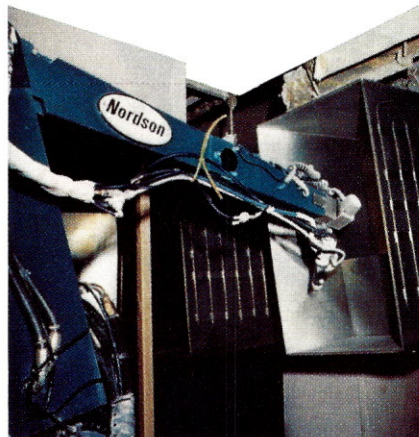
Nordson coating robots are dressing up an increasingly broad range of products—faster, safer, better and more economically.

Typical are pickup truck boxes rolling off the assembly lines of a major Ontario, Canada, automotive manufacturer. Two of our robots coat the interiors of the boxes with



better-than-ever finish quality... better-than-ever productivity... measurable savings in labor and materials.

At Franklin Manufacturing Company of St. Cloud, MN, one of the White Consolidated Industries, special clothing and equipment was needed to protect the health and safety of 15 painters spraying the inside of freezer liners because overspray bounced back out of the liners and onto the painters. Today, four Nordson robots do the same job, with



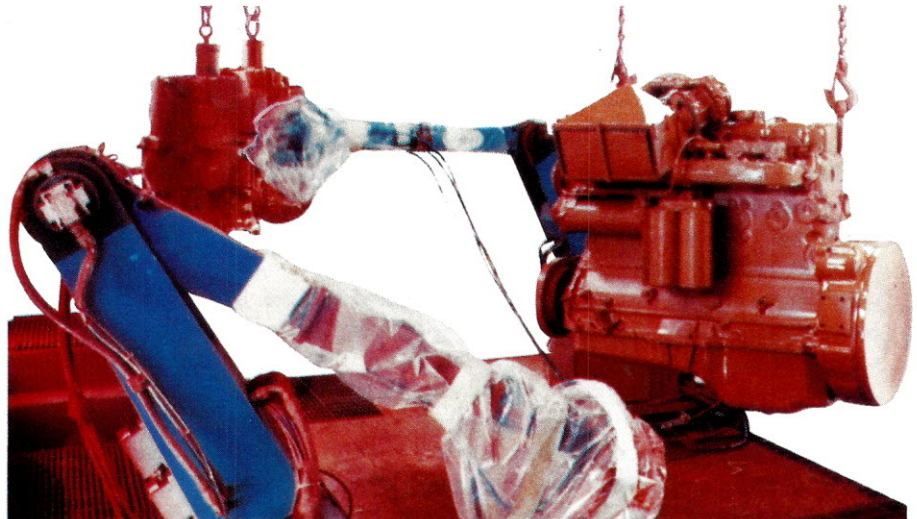
fewer rejects...no hazard to health and safety.

And at the Mack Truck manufacturing plant in Hagerstown, MD, two Nordson

paint...and without the need for manual touchup.

The burgeoning popularity of Nordson coating robots is due, in part, to the package of values that goes with them. A package that includes our worldwide network of staff engineers and technical service people. That includes tailoring a system to your exact needs, which alone could save you thousands of dollars a year. And that includes our in-depth factory training program, where you learn to work with robots.

We'd like you to visit us and see Nordson robot technology in action. To arrange your



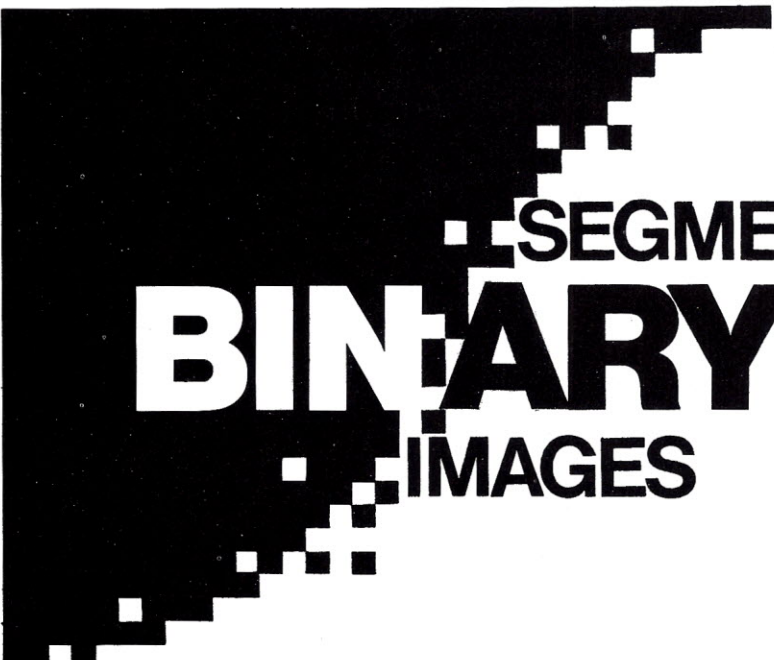
robots working in tandem apply a protective coating of paint to truck engines. They provide the same complete coverage as with the previous method of application but accomplish it using 40% less

visit, or to get more information, call toll-free 1-800-321-1414 (in Ohio, call 1-800-362-9947), Ext. 751. Or write Nordson Corporation, Robotics Division, 555 Jackson Street, Amherst, OH 44001.



Robotics Division

CIRCLE 23



SEGMENTING BINARY IMAGES

by

Robert Cunningham
Robotics Research Program
NASA Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Generalized image interpretation is still one of the most challenging and important research areas in robotics and machine intelligence. Although most current research is concerned with understanding gray scale pictures taken from an arbitrary 3D perspective, earlier results dealing with *binary* images, usually of objects confined to a plane, are being used in practical robot vision systems for a variety of applications. The main reasons for this are the compactness of representation and the simplification of the image analysis algorithms that result from structured scenes with only one bit of intensity information at each point in the picture. A 256 by 256 element binary image stored as a packed bit array requires only 8192 bytes of memory. Thus even a modest microcomputer can store a complete binary image of average spatial resolution and still have plenty of memory left over for image analysis software.

The algorithms that analyze a binary image are simpler than those which analyze gray scale images. With a binary image, the computations used in such functions as locating the edge of an object are reduced to logical operations involving only a few picture elements (pixels). As a result, binary image analysis programs are generally much faster than comparable programs that process gray scale images.

The process of producing a binary image significantly reduces the volume of data available from a standard TV camera. Of course, this reduction results in a certain amount of information loss as well, and this limits the applicability of binary image vision. However, in many robot vision tasks, particularly industrial ones such as locating parts on a conveyor belt, the designer can structure the environment to control the viewing conditions. By carefully choosing the illumination, viewing angle, and background composition, we can make it possible for vision software to distinguish target objects from the background on the basis of contrast alone. Under these circumstances, binary image vision methods can be quite effective.

One of the basic goals of a computer vision system is to determine the position and orientation of a target. A robot can use this information to guide tasks such as grasping an object, positioning a tool (as in welding or grinding), or mating parts in an assembly. In addition to locating objects, some jobs require the robot to identify what *kind* of object it is seeing or to inspect it for defects. In a controlled environment, the efficiency with which binary image vision can perform these functions has led to its use in several successful industrial systems. [refs. 1, 2]



Figure 1. Converting a gray scale image (left) to a binary image (right) is done by setting to a 1 all pixels whose brightness lies above a threshold and to 0 all those below it.

Regardless of the ultimate application, the first thing any binary image vision system must do is to partition the image into regions, or *blobs*, that correspond to objects, holes, or background in the scene. After partitioning, it can analyze the blobs to compute information needed for the task at hand.

In this article, I will present two different algorithms that produce descriptions of the outlines of blobs in a binary image. One approach, called *connectivity analysis*, is most suitable for tasks where location information and shape statistics are sufficient. The other, called *boundary tracking*, produces a precise vector description of the outlines in addition to the other results. Following this, I will discuss some of the issues involved in using the blob descriptions for object recognition, tracking moving targets, and stereo vision for rangefinding.

Getting a Binary Image

Binary images are obtained by a thresholding process that converts a continuous tone or gray scale image to one with only two "colors," black and white, much like a silhouette. There are many ways of accomplishing this conversion, either in the video input hardware or, if a gray level image is available, in software. The simplest approach, called *global thresholding*, is to compare the brightness of each pixel to a common threshold level, assigning a value of 1 (white) wherever the threshold is exceeded and 0 (black) otherwise, as shown in Figure 1. A common extension of this approach is to use two thresholds, setting to white on only those pixels whose brightness falls between the two levels.

Global thresholding with either single or dual thresholds can be built directly into the video digitizer as a one bit analog to digital converter. (See "Video Signal Input" [ref. 3] in our previous issue.) With global thresholding, you must experimentally determine the threshold level(s) that give the best results in a particular environment. If, however, a gray scale image is available, the vision software can determine the best threshold to use by looking at the intensity *histogram* of the image.

The histogram (Figure 2) shows the number of pixels with a given gray level, plotted as a function of gray level. (This is represented in our program as an array, $N(G)$, indexed by gray level.) Typically, to find a good global threshold, analyze the histogram to locate the valley between the peaks that correspond to object and background brightness levels, respectively, then set the threshold to the gray level at the valley.

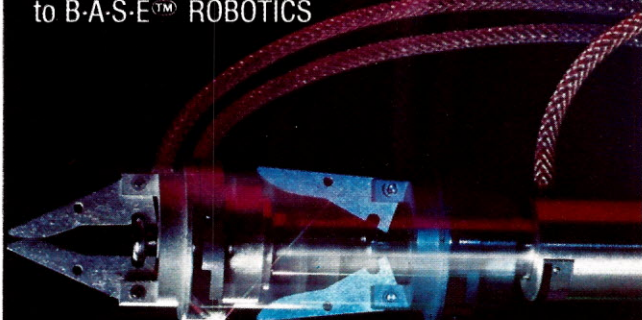
Another approach is to use not a global threshold for the entire image but a *local threshold*. The threshold used at each pixel is determined by examining the neighboring pixels in a small *window* of the image. See Weszka [ref. 4] for a more complete discussion of threshold selection techniques.

The Blob-Finding Algorithms

Assuming we have acquired a binary image in a two-dimensional bit array, let's now consider how to describe its contents. Formally, a blob is a simply connected cluster of pixels which are all the same color (i.e., black or white). Since a picture may be composed of a number of blobs, including one for the background, we can most easily

Robotics on the move

"X" Axis Transporter...
adds another dimension
to B-A-S-E™ ROBOTICS



Send for product catalogue.

MACK Corporation

3695 E. Industrial Drive, Flagstaff, AZ 86001 / (602) 526-1120
B-A-S-E™ is a Mack Corporation trademark for Basic Automation Systems Elements

CIRCLE 17

represent it as a linked list of blob descriptors. Descriptors are records that contain everything we know about a blob: its area, centroid, perimeter, number or holes, color, and possibly other shape or type information. In addition, there are links which superimpose a hierarchical structure on the blob list to represent *nesting* relationships. These relationships are described by the words *parent*, *child*, and *sibling*.

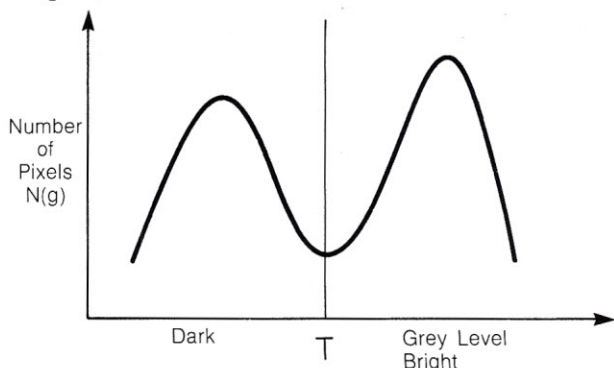


Figure 2. The histogram above represents an image with bright objects on a dark background similar to Figure 1. One method of selecting a threshold to obtain a binary image is to analyze the shape of the histogram, setting the threshold at the gray level corresponding to the valley (local minimum) which separate object pixels and background pixels.

In Figure 3, blob 1 has two children, blobs 2 and 4; the parent of blobs 2 and 4 is blob 1; and blobs 2 and 4 are siblings. Including a descriptor for the background in the blob list is convenient because it unifies the nesting relationships: blobs 1 and 3 are siblings whose common parent is the background. The parent-child relationship applies to *immediate* containment only, so that blobs 2 and 4 are not children of the background. Note that in a binary image, if two blobs are adjacent (touching), then one must contain the other, since adjacent blobs must have different colors.

Our first method of finding blobs and building the blob list description of an image is *connectivity analysis*. As its name suggests, this algorithm examines the local connectivity around each pixel of the image, building up individual blob descriptions on a pixel-by-pixel basis. Our second method is *boundary tracking*. This algorithm infers the connectivity of pixels by following the edge of a blob to determine its complete extent and outline. Connectivity analysis is the more elegant approach of the two in that all processing is done in a single raster scan in the image. The boundary tracker alternates between a raster scan search for new blobs and following the boundary, which requires random access to the image.

However, the boundary tracking algorithm has an important feature that connectivity analysis lacks—it produces a *chain-code* description of the boundary of each blob as one of its primary outputs. This is an encoded sequence of unit direction vectors that define the exact

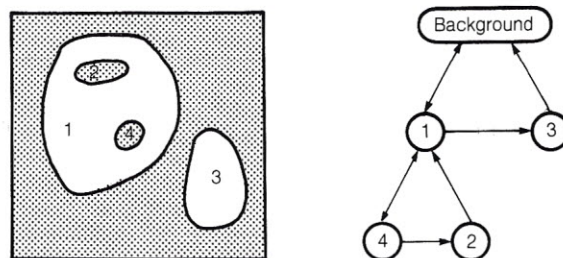


Figure 3. The blob list forms a tree-like structure to represent nesting relationships. The scene at the left contains four blobs. Blobs 1 and 3 are contained in the background. Blobs 2 and 4 are contained in blob 1. The diagram at the right indicates the linkage information stored in the blob records. The double arrow from blob 1 to blob 4, for example, indicates the *CHILD* link of blob 1 and the *PARENT* link of blob 4. The arrow from blob 4 to blob 2 is the *SIBLING* link of blob 4.

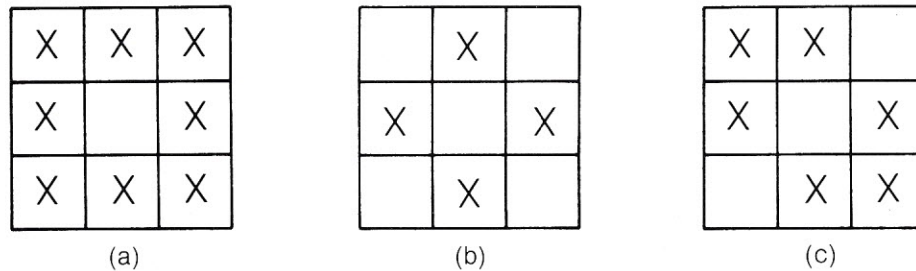


Figure 4. The x's indicate which pixels in a 3 by 3 neighborhood are connected to the center pixel using (a) 8-connectivity, (b) 4-connectivity, and (c) 6-connectivity.

shape of the blob's outline. (See the article "Chain-Code" [ref. 5] in our previous issue) This important product can make the boundary tracking algorithm the most useful of the two partitioning methods for applications that require an explicit shape analysis of object boundaries.

Before going into the details of the two approaches we must consider some problems with the definition of connectivity that impact both algorithms. Adjacent pixels are *connected* if they are the same color, but we need a convention to specify *which* pixels are adjacent. The two most natural conventions in a rectangular grid are 4-connectivity and 8-connectivity. (See Figure 4) 4-connectivity allows a pixel to be connected to any of its four nearest neighbors in the cardinal directions—up, down, left, and right. 8-connectivity allows it to be connected to all eight of its immediate neighbors by including the diagonal directions as well. This is where a problem can arise: if, for example, we use 8-connectivity to cluster pixels into white blobs, then we must use 4-connectivity to define black blobs, or *vice versa*. This is necessary to avoid a paradox that arises in images where a blob meets itself at a single point, such as point A in Figure 5a or 5b.

If all blobs are 8-connected, then from the object's point of view, the central black square is disconnected from the surrounding background and is thus a hole in the object. However, from the central square's point of view, it is connected to the background and thus is not a hole. If all blobs are 4-connected, the object views the square as part of the background while the square views itself as a hole in the object. Either way, it is impossible to generate a consistent hierarchical blob description of the scene.

One solution to this problem is to adopt a double standard which applies 4-connectivity to one color and 8-connectivity to the other. This complicates the blob finding algorithms to a certain extent, but not prohibitively so. Another solution is to adopt a 6-connectivity convention [2] which treats black and white pixels equally and avoids the paradox of Figure 5.

Unfortunately, however, 6-connectivity is not a perfect solution either. In Figure 5a the central square is 6-connected to the background, while in Figure 5b, which depicts the same object rotated 90°, the square is not connected to the background. Thus, while any image may be described consistently, the description of a particular object may vary from scene to scene. Actually, due to

quantization errors in digitized images, it is unlikely that an object will consistently appear as only one of the Figures 4a or 4b. Instead, it is more likely that pixels in the neighborhood of point A will appear as either black or white depending upon the location and orientation of the object in a particular scene. The object could appear as it does in Figures 5c or 5d as well, so that we can't expect a consistent description of such an object from one scene to the next regardless of the connectivity convention we use. We are therefore left with an essentially arbitrary decision as to which connectivity convention to adopt. In the interests of simplifying the algorithms presented here, I consistently use 6-connectivity. Now, with that out of the way, let's take a look at the blob-finding algorithms.

The blob descriptor record used by the algorithms is shown in Figure 6. The fields for the hierarchical links and the statistical features are common to both the connectivity analysis and the boundary tracking algorithms. The

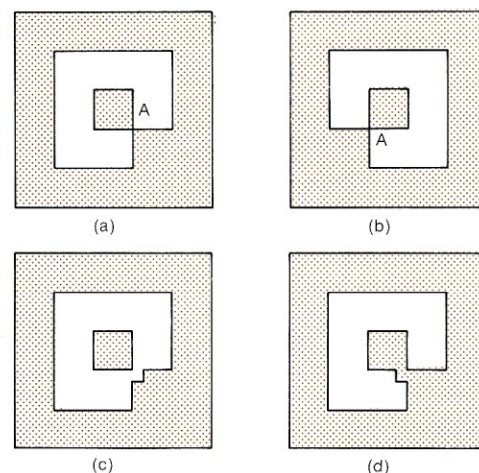


Figure 5. The blob in (a) illustrates the connectivity paradox that arises if both black and white pixels are exclusively 8-connected or 4-connected. An unambiguous description of the image, in which (a) is topologically equivalent to (c), is obtained using 6-connectivity. Note that if the object is rotated 90 degrees as in (b), the central square is a hole, topologically equivalent to (d). However, as illustrated in (c) and (d), the number or holes in this object is an unreliable feature since fluctuations in pixel values around point A due to noise and quantization errors make it unlikely that the object will be imaged consistently as in (a) or (b).

fields FLINK and RLINK are used to link the blob records into a doubly linked circular list that simplifies searching the entire collection of blob records. They are needed by the connectivity analysis algorithm, since it must insert or delete records in the list at random. In the boundary tracking algorithm, blob records are allocated sequentially as each blob is encountered so these fields are not needed. The boundary tracking algorithm has one extra field, CCB, which points to a chain-code block that contains the chain-code representation of the blob's boundary.*

As we shall see, both algorithms need a way to tell which blob a previously processed pixel has been assigned to. This information cannot be stored in the image array itself, since there is only one bit per pixel. Instead, a program must use some other representation, one that can be indexed by the pixel coordinates without requiring as much room as an entire image. One compact way of representing this knowledge is called *run-length encoding*. In the coding scheme, used by the connectivity analysis algorithm, each unbroken run of 0's or 1's in a raster line of the image is described by a record that tells its starting (leftmost) column, its length, and which blob the run belongs to. Each line of the picture can be described by a list of these run-length records.

Fields in Blob Record

FLINK	sequential	used by
RLINK	links	connectivity only
PARENT		
CHILD	hierarchical	
SIBLING	links	
AREA		
Sum of I		
Sum of J		
Sum of I ²		
Sum of J ²		
Sum of IJ	location,	used by
PERIMETER	shape,	both
IMIN	and size	
JMIN		algorithms
IMAX		
JMAX		
NHOLES	— number of holes	
CCB	— chain code block (boundary tracker only)	

Figure 6. The contents of a blob record, sharing those fields unique to either the connectivity analysis algorithm or the boundary tracking algorithm, and the fields common to both algorithms.

*Milgram [6] describes an algorithm that determines chain-code boundary descriptions from a single raster scan of the image. It requires somewhat complex data structures since the boundary is constructed in several pieces.

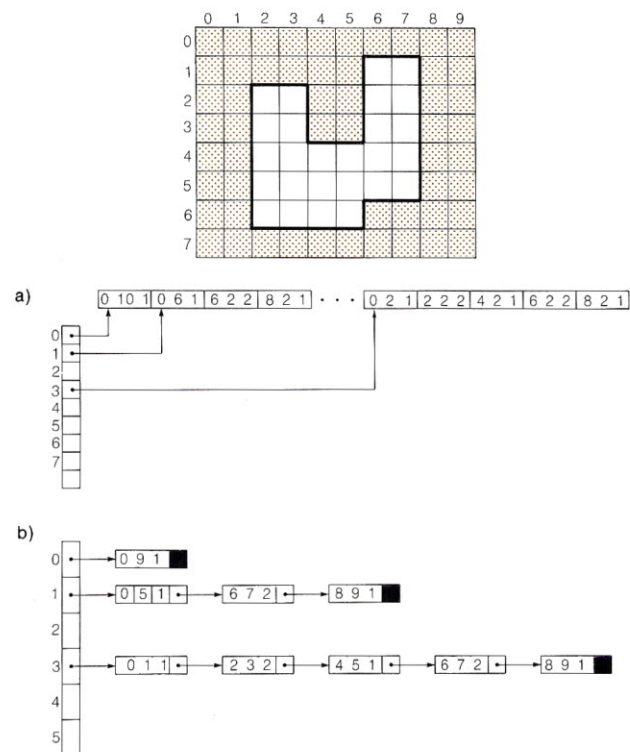


Figure 7. The blob finding algorithms generate a run-length list which encodes each run of consecutive 0's or 1's on a line as a three word record. The run-length list data structure for the connectivity analysis algorithm is shown in (a). Assuming the background is blob 1 and the object is blob 2, the three numbers in each record are interpreted as the start of a run, the number of pixels in the run, and number of the blob these pixels belong to, respectively. Part (b) shows the data structure for the boundary tracking algorithm. In this case, the second word is the right end of the run rather than its length.

As with the blob list, the structure of the run-length list differs in the two algorithms. In the connectivity analysis algorithm, run-length records are allocated sequentially as the image is scanned. In boundary tracking, run-length records are inserted and updated at random, so a linked list structure is necessary; it is also easier for this algorithm to keep the ending column of the run in the record instead of its length. In both cases, a table of pointers to the first record of each line is maintained. Figure 7 shows the run-length list structure for both algorithms. It's interesting to note that since the connectivity analysis algorithm operates strictly in a raster scan, it is very easy to modify it to accept a run-length description of the image as its only input. In this case the blob assignment fields would initially indicate only the color of each run.

Some final comments about the data structures: It is convenient to assume that there is a margin of at least one background pixel around the entire image. This greatly simplifies the algorithms, since there are then no special boundary conditions to consider. We are now ready for the details of the algorithms. In the following discussion,

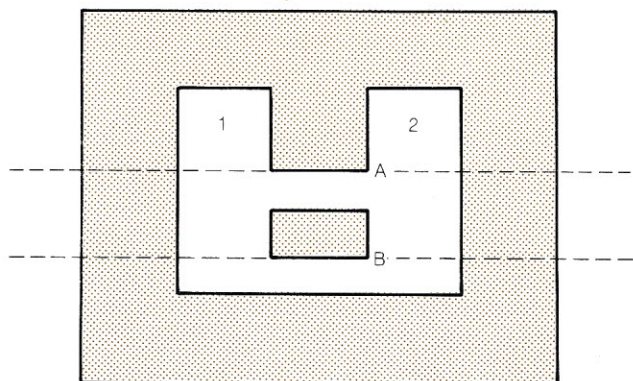


Figure 9. In the connectivity analysis algorithm, separate blob records are maintained for the components labeled 1 and 2 until the raster scan reaches the point labeled A, at which point the two records merge into one. When the scan reaches point B, it discovers that the same blob has surrounded an interior hole.

the image size is NROW lines by NCOL columns. Pixels are addressed by $(I,J) = (\text{column}, \text{line})$ where I can vary from 0 to NCOL-1 and J from 0 to NROW-1. The upper left corner of the image is $(0,0)$.

Connectivity Analysis

A version of the connectivity analysis algorithm written in PASCAL is given in Figure 8. Blob records are allocated from an array B, indexed by blob number, and run-length records from array R. Array RP, indexed by image line number, holds pointers to the first run-length record of each line. The constants define the particular limits of my system—yours may differ. Procedure GETLIN is the interface to the bit array of the binary image. I don't show the actual method it used to unpack an image line into an integer array, since it is highly machine dependent for optimum efficiency.

The connectivity analysis algorithm scans the image from left to right, top to bottom, updating the descriptions of each blob which intersects the current scan line. At the end of each run of 0's or 1's, the run-length list is updated and blob statistics are accumulated. If any pixel of the run just completed is 6-connected to a pixel of the same color on the previous line, an existing blob is extended to include this run. Otherwise, a new blob is record is allocated. Each blob is assigned a unique number. Pixels belonging to the blob are labeled with this number in the run-length list.

A blob may initially have more than one label associated with it, as shown in Figure 9. It is not until the scan reaches point A that the two components labeled 1 and 2 are recognized as being part of the same blob. At this point they are merged—the record for blob 2 is released after combining the statistics of blobs 1 and 2.

However, there are still references to blob 2 in the run-length list, so we need a method of making labels 1 and 2 equivalent. The method I use involves a table of blob

pointers (BP in the program) which is indexed by blob number. If the entry in the table for a given blob number is positive, then it is a valid pointer to an active blob record. If the entry is negative, then the original blob record has been merged and released, and the corresponding positive value of the entry is the number of the blob that was merged with the original blob. It may be necessary to chain through several locations if more than one merge has occurred in the same blob.

The hierarchical relationships are updated whenever a blob is complete, as detected by the surrounding blob closing on itself. When this occurs, as shown at point B in Figure 9, the surrounded blob is inserted into the list of children of the current blob by manipulating the PARENT, CHILD, and SIBLING pointers in the two blob records.

The program could compute the moments and other blob statistics on a pixel-by-pixel basis. However, it's better to update blob statistics at the end of each run since this requires far fewer computations. Also, new blob records are not allocated until the first run of a new blob has ended. Whenever a new blob record is allocated

Updating Blob Statistics

Area, Sums of I , J , I^2 , J^2 , and IJ

Since the I values of a run of L pixels starting with I_0 are $I_0+0, I_0+1, \dots, I_0+(L-1)$, and since

$$\sum_{k=0}^{L-1} k = L(L-1)/2,$$

then the contribution of the run to the sum of I is $L(I_0+(L-1)/2)$.

Similarly, since $\sum_{k=0}^{L-1} k^2 = L(L-1)(2L-1)/6$,

and $(I+k)^2 = I^2 + 2Ik + k^2$, then the contribution of the run to the sum of I^2 is

$$\sum_{k=0}^{L-1} (I_0+k)^2 = L[I_0^2 + I_0(L-1) + (L-1)(2L-1)/6].$$

Since J is constant over the run, the contribution of the run to the sum of IJ is $LJ(I_0+(L-1)/2)$, the sum of J is incremented by LJ and the sum of J^2 is incremented by LJ^2 .

Figure 10. The formulae for updating moments at the end of each run in the connectivity analysis algorithm.

(procedure ALLOC in the program), its moments, perimeter, and hole count are zeroed. The blob's minimum enclosing rectangle is initialized by setting JMIN to J (the current line number), IMIN to the starting column number of the run, and IMAX to the last column in the run.

As new runs are added to the blob on successive lines, the statistics have to be updated (procedure UPDATE). The moments are incremented using the formulae given in Figure 10. These take advantage of the fact that J, the line

number, is constant over the length of the run, and, since the run covers consecutive columns, the moments involving I can be collected using the closed forms for sums of powers of consecutive integers. The enclosing rectangle is updated by checking if the left or right endpoints of the new run exceed the old values of IMIN or IMAX, respectively. Since the blob grows steadily downward, JMAX is simply set to J, and there is no need to change JMIN. JMIN is only updated when two blobs merge, in which case the other extreme must be checked and changed as well to accurately represent the extent of the resulting merged blob. The perimeter is incremented by two to account for the vertical boundary segments at each end of the run. The contributions to the perimeter of the horizontal boundary segments are handled specially, as I will describe shortly.

The various actions described above take place whenever there is a transition from black to white or from white to black in either the current line or the previous line. To detect these transitions, the image is scanned with what amounts to a 2 by 2 "window" into the image whose lower right corner is always the current pixel. Thus, the previous pixel on the same line and the two corresponding pixels on the previous line make up the other elements in the window. Since each pixel can be colored either black or white (0 or 1), there are 2^4 or 16 possible patterns that can appear in the 2×2 window. These 16 patterns or window states are shown in Figure 11.

These window states are numbered by treating the values of the pixels in the window as a four bit binary number, according to the formula shown in the figure. Although there are 16 distinct states, there are actually only eight cases to consider. Since we want to treat transitions from black to white the same as those from white to black, the binary complement of a state number (1's and 0's interchanged) represents the same transition and should be handled identically. Of the eight remaining cases, two (0,15 and 5,10) require no action at all, since there is no horizontal transition. Let's now consider the other six cases in detail.

In the discussion below, as well as in the program, I is the position of the scan corresponding to the column coordinate of pixel A in the 2×2 window. CURREG and REGABV are initially pointers to the blob records corresponding to the pixels C and D, respectively. At the beginning of each line, CURREG and REGABV are both set to the background color (since there is one pixel margin of background around the entire image) and the scan actually starts in the second column from the left. In the program, variable WSTATE, derived using the state

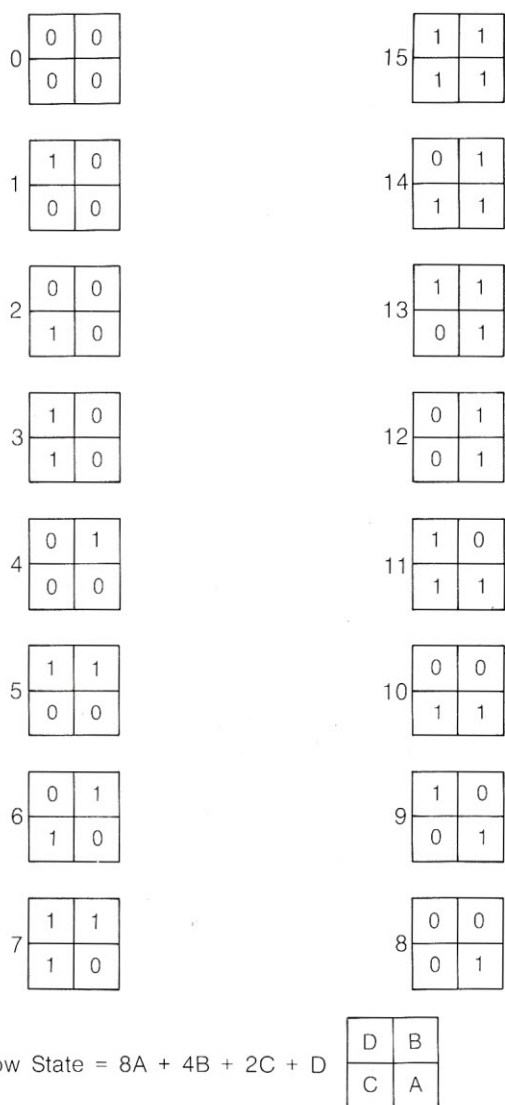


Figure 11. The 16 possible configurations of 0's and 1's are shown above, numbered according to the convention shown at the bottom. Note that they are arranged in complementary pairs.

labeling formula in Figure 11, identifies which window state has occurred. (The DIV operator used in the code produces the same results due to the iteration.) Here, then, are the six transitions we still need to consider:

Case 7,8: This is possibly the start of a new blob. Set the variable NEWFLG to true. It is also the beginning of a new horizontal boundary segment (which will contribute to the perimeter) and a new run. Save the I coordinate in the variable LFTEND for later use in the update. Update the statistics of the run just completed.

Case 4,11: Although this is not the start of a new run, it does mark the beginning of a new horizontal boundary segment—therefore LFTEND is again set to I. Set REGABV to the blob assignment of pixel B, obtained by looking it up in the run-length list.

Case 3,12: Update the run-length list and blob statistics for the run just ended. REGABV and CURREG are both set to the blob assignment of pixel B.

Case 2,13: If NEWFLG is true, allocate a new blob record for the run just completed. Set CURREG to the number of this record and NEWFLG to false. Update the statistics of the new blob, including the perimeter, which is incremented (from zero) by the amount I-LFTEND. Set CURREG to REGABV.

Case 6,9: If NEWFLG is true, allocate a new blob and update it as in Case 2,13. Set CURREG to the current value of REGABV to the blob label of pixel B.

Case 1,14: There are three possibilities here. If NEWFLG is true, then what was originally assumed to be a new blob has merged with an existing blob. If NEWFLG is false, then either two distinct blobs have merged or a blob has closed on itself surrounding the blob containing pixel D. Save REGABV (pixel D's blob) in variable HOLREG, and then set REGABV to the label of pixel B. If NEWFLG is true, then set CURREG to the new REGABV and NEWFLG to false. This will include the new run in the old blob.

If NEWFLG is false and CURREG is *not* equal to the new REGABV, then merge the two blobs by releasing REGABV, deleting it from the sequential blob list and combining both blobs' statistics into CURREG. Update the blob number table BP[REGABV] to make REGABV equivalent to CURREG. If, however, NEWFLG is false and CURREG is equal to REGABV, then add HOLREG to the list of children of CURREG, since it is indeed a hole in CURREG (increment the hole count as well).

Regardless of NEWFLG, this case also terminates a horizontal boundary segment of HOLREG and CURREG. If HOLREG is not really a hole (CURREG not equal to REGABV), then subtract the perimeter of HOLREG from that of CURREG (since we only want the outer perimeter) and increment the perimeter of HOLREG by I-LFTEND.

To perform a connectivity analysis of a scene, the computer program calls the procedure BLOBSCAN with parameters that define the upper left and lower right corners of a rectangular window into the image bit array. When BLOBSCAN returns, the blob records can be examined by following the linkages from the background blob record, B[1], either sequentially (following FLINK or RLINK) or hierarchically (the CHILD field points to the first child blob, whose SIBLING link points to the next, etc.).

Boundary Tracking

The boundary tracking algorithm traverses the boundary of each blob in the image when it is first encountered in a raster scan of the image. As the boundary is traversed, blob statistics are computed and the run-length list is updated to incorporate the new blob. In addition, the algorithm produces a chain-code representation of the boundary [5]. A new blob is recognized whenever the raster scan encounters a color transition between (I-1, J) and (I, J) which is not yet represented in the run-length list. When this occurs, the raster scan is suspended and the boundary tracker is invoked to immediately get a complete description of the new blob. When the boundary tracker has completed, the raster scan is resumed at (I+1, J). Obviously, the boundary of each blob will be encountered several times during the raster scan, but the boundary tracker is invoked only once per blob, since all encounters after the first are represented in the run-length list.

There are three distinct elements of the boundary tracking algorithm. The first is the tracking procedure itself. The second is the procedure that computes the blob statistics as the boundary is traversed. The third is the procedure that updates the run-length list. Let's look at each of these in turn.

The boundary tracker views each pixel as a unit square in the image plane, bounded by four edges. A boundary pixel is one which has one or more edges in common with a background pixel (for the remainder of this section, the word *background* is used in a generic sense to indicate the *parent* of a blob). These edges are called *boundary segments*. The boundary of a blob is a sequence of

Figure 8. This is a complete PASCAL procedure implementing the connectivity analysis procedure described in the text. The calling program is not shown. The constants at the top, defining various array lengths and the image size, are representative. The actual values you choose will depend on your video input hardware and the particular application.

```

CONST  MAXBLOB=25;  MAXRUN=1000;  MAXNUM=200;
       NROW=240;  MAXROW=239;  NCOL=192;  MAXCOL=191;

TYPE  BLOB=RECORD
      FLINK,RLINK,PARENT,CHILD,SIBLING: INTEGER;
      COLOR,NUMBER,PERIM,NHOLES: INTEGER;
      IMIN,IMAX,JMIN,JMAX: INTEGER;
      AREA,SUMI,SUMJ,SUMI2,SUMIJ,SUMJ2: REAL
    END;

      RUN=RECORD
      START,LENGTH,NUMBER: INTEGER
    END;

      LINE=ARRAY [0..MAXCOL] OF INTEGER;

VAR  B: ARRAY [1..MAXBLOB] OF BLOB;
     BP: ARRAY [1..MAXNUM] OF INTEGER;
     BN: INTEGER;
     R: ARRAY [1..MAXRUN] OF RUN;
     RP: ARRAY [0..MAXROW] OF INTEGER;

PROCEDURE QUIT; (* this is a dummy — its supposed to be an error exit *)

PROCEDURE GETLIN(J:INTEGER; VAR L: LINE);
(* this is implementation dependent — it stores line J in array L *)

FUNCTION LOOKUP(N: INTEGER): INTEGER;
BEGIN
  WHILE BP[N] < 0 DO N:=BP[N];
  LOOKUP:=N;
END;

PROCEDURE BLOBSCAN(IL,JL,IR,JR: INTEGER);

VAR  I,J,WSTATE,CRUN,PRUN,NEWFLAG,LEFTEND: INTEGER;
     CURREG,REGABV,HOLREG: INTEGER;
     FREE,AVAIL: INTEGER;
     CLINE,PLINE: LINE;
     TMP,TMP2: INTEGER;

PROCEDURE INIBLOB(N,L,R,J: INTEGER);
BEGIN
  IF BN > MAXNUM THEN
    BEGIN WRITE('**BLOBNUM OVERFLOW'); QUIT END
  ELSE
    BEGIN
      BP[BN]:=N;  B[N].NUMBER:=BN;  BN:=BN+1
    END;
    B[N].PARENT:=1;
    B[N].CHILD:=1;
    B[N].SIBLING:=1;
    B[N].COLOR:=CLINE[R];
    B[N].AREA:=0;
    B[N].SUMI:=0;
    B[N].SUMJ:=0;
    B[N].SUMI2:=0;
    B[N].SUMIJ:=0;
    B[N].SUMJ2:=0;
    B[N].IMIN:=L;
    B[N].IMAX:=R;
    B[N].JMIN:=J;
    B[N].JMAX:=J;
  END;

PROCEDURE ALLOC;
BEGIN
  NEWFLAG:=0;
  IF AVAIL < 0 THEN
    BEGIN
      IF FREE > MAXBLOB THEN
        BEGIN WRITE('**BLOBLIST OVERFLOW'); QUIT END
      ELSE
        BEGIN
          CURREG:=FREE;
          FREE:=FREE+1;
        END
      ELSE
        BEGIN
          CURREG:=AVAIL;
          AVAIL:=B[AVAIL].FLINK;
        END;
      INIBLOB(CURREG,R[CRUN].START,I-1,J);
      TMP:=B[REGABV].FLINK;
      B[REGABV].FLINK:=CURREG;
      B[CURREG].FLINK:=TMP;
      B[TMP].RLINK:=CURREG;
      B[CURREG].RLINK:=REGABV;
    END;

PROCEDURE UPDATE(EOL: INTEGER);
VAR  L,K: REAL;
BEGIN
  L:=I - R[CRUN].START;
  R[CRUN].LENGTH:=I - R[CRUN].START;
  R[CRUN].NUMBER:=B[CURREG].NUMBER;
  B[CURREG].PERIM:=B[CURREG].PERIM + 2;
  K:=R[CRUN].START;
  B[CURREG].AREA:=B[CURREG].AREA + L;
  B[CURREG].SUMI:=B[CURREG].SUMI + L*((L-1)/2 + K);
  B[CURREG].SUMJ:=B[CURREG].SUMJ + L*J;
  B[CURREG].SUMI2:=B[CURREG].SUMI2 + L*((L-1)*K + (2*L-1)/6 + K*K);
  B[CURREG].SUMIJ:=B[CURREG].SUMIJ + L*((L-1)/2 + K)*J;
  B[CURREG].SUMJ2:=B[CURREG].SUMJ2 + L*J*J;

  IF R[CRUN].START < B[CURREG].IMIN THEN  B[CURREG].IMIN := R[CRUN].START;
  IF I-1 > B[CURREG].IMAX THEN  B[CURREG].IMAX := I-1;
  B[CURREG].JMAX:=J;
  CRUN:=CRUN+1;
  IF CRUN > MAXRUN THEN
    BEGIN WRITE('**RUNLIST OVERFLOW'); QUIT END;
  IF EOL = 0 THEN
    R[CRUN].START:=I
  ELSE
    BEGIN
      R[CRUN].START:=IL;
      PRUN:=PRUN+1;
    END;
  END;

PROCEDURE STATE7;
BEGIN
  UPDATE(0);
  LEFTEND:=I;
  NEWFLAG:=1
END;

PROCEDURE STATE4;
BEGIN
  LEFTEND:=I;
  PRUN:=PRUN+1;
  REGABV:=LOOKUP(R[PRUN].NUMBER);
END;

PROCEDURE STATE3;
BEGIN
  UPDATE(0);
  PRUN:=PRUN+1;
  REGABV:=LOOKUP(R[PRUN].NUMBER);
  CURREG:=REGABV
END;

PROCEDURE STATE2;
BEGIN
  IF NEWFLAG <> 0 THEN ALLOC;
  UPDATE(0);
  B[CURREG].PERIM:=B[CURREG].PERIM + I - LEFTEND;
  B[REGABV].PERIM:=B[REGABV].PERIM + I - LEFTEND;
  CURREG:=REGABV
  PRUN:=PRUN+1;
  REGABV:=LOOKUP(R[PRUN].NUMBER);
  LEFTEND:=I;
END;

PROCEDURE STATE6;
BEGIN
  IF NEWFLAG <> 0 THEN ALLOC;
  UPDATE(0);
  B[CURREG].PERIM:=B[CURREG].PERIM + I - LEFTEND;
  B[REGABV].PERIM:=B[REGABV].PERIM + I - LEFTEND;
  CURREG:=REGABV;
  PRUN:=PRUN+1;
  REGABV:=LOOKUP(R[PRUN].NUMBER);
  LEFTEND:=I;
END;

PROCEDURE STATE1;
BEGIN
  HOLREG:=REGABV;
  PRUN:=PRUN+1;
  REGABV:=LOOKUP(R[PRUN].NUMBER);
  IF NEWFLAG <> 0 THEN
    BEGIN
      NEWFLAG:=0;
      CURREG:=REGABV;
      B[CURREG].PERIM:=B[CURREG].PERIM + I - LEFTEND;
    END
  ELSE IF CURREG = REGABV THEN
    BEGIN
      TMP:=B[CURREG].CHILD;
      B[HOLREG].SIBLING:=TMP;
      B[HOLREG].PARENT:=CURREG;
      B[CURREG].CHILD:=HOLREG;
      B[CURREG].NHOLES:=B[CURREG].NHOLES + 1;
      B[CURREG].PERIM:=B[CURREG].PERIM - B[HOLREG].PERIM
    END
  ELSE
    BEGIN
      B[CURREG].PERIM:=B[CURREG].PERIM + B[REGABV].PERIM + I - LEFTEND;
      IF B[REGABV].IMIN < B[CURREG].IMIN THEN
        B[CURREG].IMIN:=B[REGABV].IMIN;
      IF B[REGABV].IMAX > B[CURREG].IMAX THEN
        B[CURREG].IMAX:=B[REGABV].IMAX;
      IF B[REGABV].JMIN < B[CURREG].JMIN THEN
        B[CURREG].JMIN:=B[REGABV].JMIN;
      B[CURREG].AREA:=B[CURREG].AREA + B[REGABV].AREA;
      B[CURREG].SUMI:=B[CURREG].SUMI + B[REGABV].SUMI;
      B[CURREG].SUMJ:=B[CURREG].SUMJ + B[REGABV].SUMJ;
      B[CURREG].SUMI2:=B[CURREG].SUMI2 + B[REGABV].SUMI2;
      B[CURREG].SUMIJ:=B[CURREG].SUMIJ + B[REGABV].SUMIJ;
      B[CURREG].SUMJ2:=B[CURREG].SUMJ2 + B[REGABV].SUMJ2;
      IF B[REGABV].NHOLES <> 0 THEN
        BEGIN
          B[CURREG].NHOLES:=B[CURREG].NHOLES + B[REGABV].NHOLES;
          TMP:=B[REGABV].CHILD;
          WHILE TMP > 0 DO
            BEGIN
              B[TMP].PARENT:=CURREG;
              TMP2:=TMP;
              TMP:=B[TMP2].SIBLING;
            END;
          B[TMP2].SIBLING:=B[CURREG].CHILD;
          B[CURREG].CHILD:=B[REGABV].CHILD;
        END;
    END;

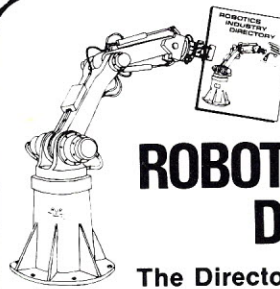
```



```

B[B[REGABV].RLINK].FLINK:=B[REGABV].FLINK;
B[B[REGABV].FLINK].RLINK:=B[REGABV].RLINK;
BP[B[REGABV].NUMBER]:= -B[CURREG].NUMBER;
B[REGABV].FLINK:=AVAIL;
AVAIL:=REGABV;
REGABV:=CURREG;
END;
B[HOLREG].PERIM:=B[HOLREG].PERIM + I - LEFTEND;
END;
BEGIN (* BLOBSCAN *)
BN:=1;
FREE:=2;
AVAIL:=1;
FOR I:=1L TO IR DO PLINE[I]:=0;
INIBLOB(1,0,MAXCOL,0);
B[1].FLINK:=1;
B[1].RLINK:=1;
RP[JL]:=1;
R[1].START:=1L;
R[1].LENGTH:=NCOL;
R[1].NUMBER:=1;
PRUN:=1;
R[2].START:=1L;
CRUN:=2;
FOR J:= JL+1 TO JR DO
  BEGIN
    NEWFLAG:=0;
    WSTATE:=0;
    RP[J]:=CRUN;
    GETLIN(J,CLINE);
    REGABV:=LOOKUP[R[PRUN].NUMBER];
    CURREG:=REGABV;
    FOR I:= 1L+1 TO IR DO
      BEGIN
        WSTATE:= (WSTATE DIV 4) + 4*PLINE[I] + 8*CLINE[I];
        PLINE[I]:=CLINE[I];
        CASE WSTATE OF
          7,8: STATE7;
          4,11: STATE4;
          3,12: STATE3;
          2,13: STATE2;
          6,9: STATE6;
          1,14: STATE1
        END;
      END;
    END;
    UPDATE(1);
  END;
END;
END;

```



ROBOTICS AGE
presents

The Annual 1981

ROBOTICS INDUSTRY DIRECTORY

The Directory contains comprehensive listings of:

- Robot Companies that manufacture complete systems
- Related Manufacturers of robot modules and components
- Institutions, laboratories, and universities performing research related to Robotics
- Consulting firms with expertise in the field

Each entry will consist of a brief description of the product line, activity, service, etc., with the name, address, and phone number of a knowledgeable representative.

Who needs the Robotics Industry Directory?

- **Engineering Personnel:** technical proposal preparation, technology surveys, systems integration, cost estimation.
- **Management:** consulting services, feasibility studies, in-house requirements.
- **Procurements:** bid package preparation, general equipment specification, sources, costing.
- **Marketing:** product planning, market surveys, sales projections, competitive appraisal.

Order Today — Only \$24.95

(Canada \$26.45 US, other Foreign \$31.45 US)

Robotics Industry Directory, PO Box 512, Tujunga, CA 91042

THE COMPUTER

GRAPHICS

SOFTWARE

NEWS

You've needed to know more. Now you can read concise reviews of Industrial and Public Domain software packages for Computer Graphics.

Subscribe to the software user's newsletter.

Fifty Dollars for Six Issues per Year

5857 South Gessner, Ste. 401, Houston, Texas 77036

directed boundary segments represented using a four-direction chain-coding scheme. The four directions are left, up, right and down, numbered 0, 1, 2 and 3, respectively (see Figure 12a). Note that in the eight-direction chain-code (Figure 12b), these directions are labeled 0, 2, 4 and 6, respectively.

Each boundary segment in the chain code representation corresponds to one edge of a boundary pixel. Thus, associated with the boundary as we have defined it is a sequence of boundary pixels. An alternative way to represent the boundary is with a chain-coded list of vectors that connect adjacent boundary pixels. This requires the eight direction chain-coding scheme. Figure 13 shows both boundary representations for the same blob. While the two representations are equivalent in the sense that one can be uniquely derived from the other, there are some differences. First, some of the statistics, particularly the area and perimeter, have different values, depending on which representation is used. This is not a serious problem since the values will be consistent if the same representation is always used. Note that the statistics obtained from the boundary segment representation are identical to those obtained from the connectivity

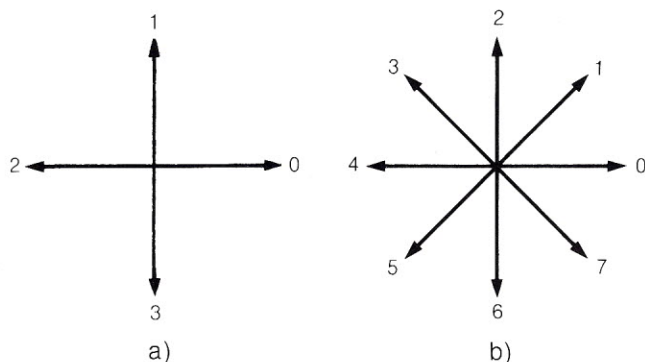


Figure 12. The boundary tracking algorithm uses a four direction chain-coding scheme with the four directions numbered as shown above in (a). The numbering for eight direction chain-coding is shown for comparison in (b).

analysis algorithm described earlier. The second difference is illustrated in Figure 14. In this case, the blob has a component that is only a single pixel wide. In the boundary pixel representation, we cannot obtain a closed boundary unless we allow the boundary to backtrack over itself. Here, we avoid this situation by using four direction chain-code.

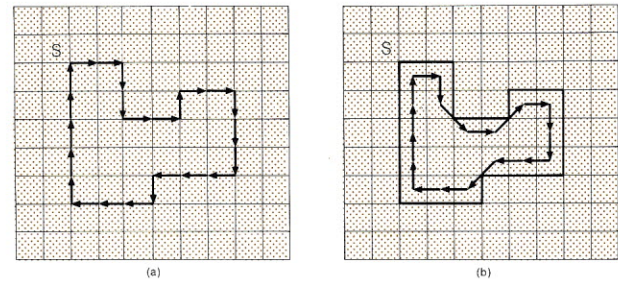


Figure 13. Starting at the point labeled S and traveling clockwise, the four direction chain-code for the object in (a) obtained by following pixel edges is 003300100333222322211111. The eight direction chain-code for the same object obtained by connecting pixel centers in (b) is 06701066445442222. Notice that the area and perimeter of the object derived from the two chain-code representations is different.

The boundary tracking procedure operates as follows. Suppose the raster scan has just encountered a new blob at (I,J). This becomes the first boundary pixel and the coordinates are saved in variables (I₀,J₀). The chain code list is initialized with a "right" (0) boundary segment code corresponding to the top edge of the initial pixel, and this value is also stored in the variable CC. The procedure BTRACK, with alterable (VAR) arguments CC, I, and J, is now called repeatedly to locate successive boundary segments around the blob. Each call to BTRACK replaces the old values of its arguments with new ones. It returns in CC the next boundary code, which is added to the chain code list, and in I and J the coordinates of the next boundary pixel. The tracking procedure terminates when CC=0 and (I,J)=(I₀,J₀). The field CCB in the blob record is set to point to the completed chain-code list. (The chain-code is usually stored in packed form, i.e., four codes per byte for four direction code.)

The boundary traversal is clockwise with respect to the

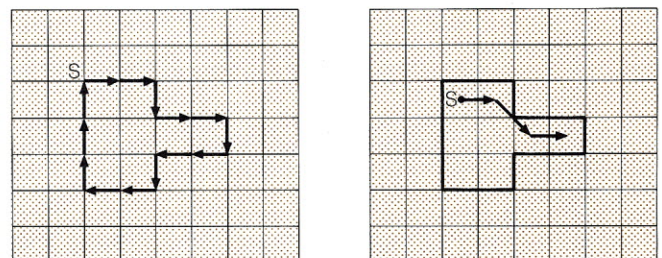


Figure 14. The boundary tracking algorithm uses four direction chain-coding (left) to avoid situations like that shown at the right which may occur if eight direction chain-coding is used. In the latter example the boundary must retrace itself to form closed loop.

interior of the blob, or, equivalently, the blob is always on the righthand side of a directed boundary segment. At each step, BTRACK decides whether to turn to the left or right or to continue straight ahead. This requires examining the two pixels, labeled A and B in Figure 15, that lie ahead and on either side of the current boundary segment. The figure assumes that a white blob is being tracked.

The 6-connectivity convention results in slightly different

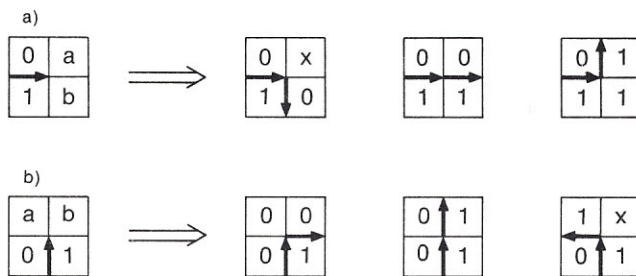


Figure 15. The rules for determining the next boundary segment in the boundary tracking algorithm depend on the colors of the pixels A and B in the 2 by 2 windows shown at the left for the cases when the last segment was "right" (a) and "up" (b). It is assumed that the object is composed of 1's. The rules for proceeding from a "left" or "down" boundary segment are obtained by rotating (a) or (b), respectively, by 180 degrees.

decision rules for vertical than for horizontal boundary segments. In all cases, if pixel A (Figure 15) is black and B is white, the boundary continues in the same direction. The differences arise in deciding to turn left or right. In the case where the current segment is horizontal (left or right), pixel A is not 6-connected to the current boundary pixel. Thus the boundary turns left only if pixels A and B are both white. If pixel B is black, then the boundary turns right regardless of the color of A.

In the case where the current segment is vertical, pixel A is 6-connected to the current boundary pixel, so the boundary turns left if pixel A is white regardless of the color of pixel B. The boundary turns right only if pixels A and B are both black. Each of these cases is shown in the figure. In general, when the boundary turns left or continues in the same direction, the next boundary pixel is A or B, respectively. When the boundary turns right, the boundary pixel remains the same as the current one. The rules for tracking the boundary of a black blob are identical, with the roles of black and white reversed.

Next, let's consider the accumulation of blob statistics as the boundary is traversed. The perimeter of the blob is

a) Moment Formulae

$$\begin{aligned} \text{Area} &:= P_1 \\ \sum I &:= P_2/2 \\ \sum J &:= P_3/2 \\ \sum I^2 &:= P_4/3 \\ \sum IJ &:= (2P_5 - P_6)/4 \\ \sum J^2 &:= P_7/3 \end{aligned}$$

b) Accumulation of P terms

If boundary code = 0 ($\rightarrow$)

$$\begin{aligned} X &:= X+1 \\ P_3 &:= P_3 - Y^2 \\ P_7 &:= P_7 - Y^3 \end{aligned}$$

If boundary code = 1 ($\uparrow$)

$$\begin{aligned} Y &:= Y+1 \\ P_1 &:= P_1 - X \\ P_2 &:= P_2 - X^2 \\ P_4 &:= P_4 - X^3 \\ P_5 &:= P_5 - X^2Y \\ P_6 &:= P_6 + X^2 \end{aligned}$$

If boundary code = 2 ($\leftarrow$)

$$\begin{aligned} X &:= X-1 \\ P_3 &:= P_3 + Y^2 \\ P_7 &:= P_7 + Y^3 \end{aligned}$$

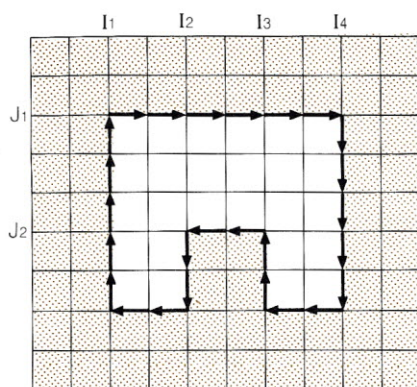
If boundary code = 3 ($\downarrow$)

$$\begin{aligned} Y &:= Y-1 \\ P_1 &:= P_1 + X \\ P_2 &:= P_2 + X^2 \\ P_4 &:= P_4 + X^3 \\ P_5 &:= P_5 + X^2Y \\ P_6 &:= P_6 + X^2 \end{aligned}$$

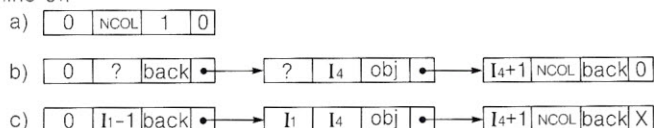
Initially, $X := I_0 - .5$ and $Y := J_0 - .5$

Figure 16. In the boundary tracking algorithm, moments are calculated by the formulas shown in (a). The terms P_1 and P_2 , etc. are accumulated according to the formulas shown in (b), where the new value of each term is computed from its previous value and the current pixel edge coordinates.

simply the length of the chain-code list and thus requires a counter which is incremented for each iteration of BTRACK. The moments (area, I , J , I^2 , J^2 , IJ) are computed according to the formulas described in [5]. Since we are using a four direction chain-code, we need only the formulas for horizontal and vertical segments. These are summarized in Figure 16.



Evolution of
line J1:



Evolution of
line J2:

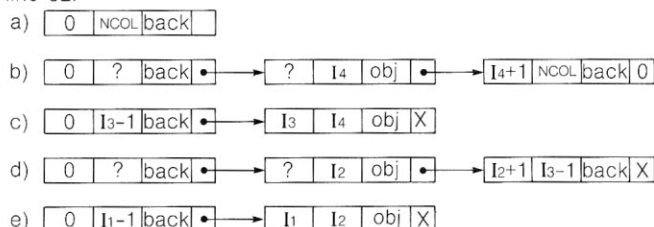


Figure 17. The evolution of the run-length list as the boundary of the object at the top is traversed is illustrated for line J1 and J2. Question marks indicate unknown values which will be filled in later when the line is crossed.

Taking I as an example, the value computed by these formulas is equivalent to summing the I coordinates of each pixel in the blob, as is done implicitly in the connectivity analysis algorithm. However, we can not use the pixel coordinates directly in the boundary traversal formulas since the boundary segments do not pass through pixel center, but rather lie on the edges of pixels. Thus the values of x and y in the formulas of Figure 16 must be the coordinates of a corner of the current boundary pixel, and are those of the form $x = I \pm .5$ and $y = J \pm .5$. Recalling that the boundary is initialized with the top edge of the first boundary pixel, the coordinates of the starting point of the boundary for the purposes of computing moments becomes $(I-.5, J-.5)$.

The enclosing rectangle represented by IMIN, JMIN, IMAX, and JMAX is initialized by setting IMIN and IMAX to I0, and JMIN and JMAX to J0. By the nature of the algorithm's raster scan search for new blobs, J0 must be the value of JMIN, so JMIN need never be changed. JMAX is updated whenever the next boundary segment is down. Likewise, IMIN or IMAX are only updated when the next boundary segment is left or right, respectively. The natural place to do this is in the procedure BTRACK

where the current value of CC, holding the code for the current boundary segment, must be tested to determine which pixels it examines to compile the next boundary segment.

Finally, the algorithm must maintain the run-length list. Each run-length record has four fields, labeled IL, IR, BLOBID, and LINK, corresponding to the left and right endpoints (column coordinates) of the run, a pointer to the blob record for the pixels in this run, and a pointer to the next run-length record for this line. The LINK field is -1 for the last record of each line. The run-length list is initialized to represent the entire image as single blob of background pixels; i.e., there is one record per line of the form $(0, NCOL-1, BACK, -1)$ where BACK is the number of the actual background blob record.

As a blob boundary is traversed, new run-length records are created for each run of pixels contained in the blob. Assuming that for some line J the blob extends from I1 to I2, the run-length list can be updated as follows:

1. Find the background record line J that contains the run, i.e., IL, I2, I2, IR.
2. Create a new run-length record $(I1, I2, NEW)$, where NEW is the current blob number. (The LINK field is set in step 4.)
3. Replace IR in the background record by I1-1 and set the LINK field of this record to point to the new record created in step 2.
4. Create a new background record $(I2+1, IR, BACK, -)$, set the LINK field of this record to the old value in the original background record, and set the link of the record created in step 2 to point to this record.

Unfortunately, the actual procedure for updating the run-length list is a little more complicated than the above description, since I1 and I2 must be determined by two separate crossings of line J in the boundary tracking procedure. Assuming that line J is crossed at I2 first, and noting that the right end of a run must be associated with a "down" boundary segment, the above procedure can be modified as follows:

1. Find the background record for line J that contains I2.
2. Create a new run-length record for the current blob $(-1, I2, NEW, -)$.
3. Replace IR in the background record by -1 and fix its LINK.
4. Create a new background record $(I2+1, IR, BACK, -)$ and set the LINKs of both new records.

The -1's in the new blob record and the original

background record indicate that the corresponding end-points are not yet known. These fields will be resolved by a later crossing of line J by an "up" segment which determines the left (right) end of the blob (background) run. This requires a different procedure that is called when line J is crossed the second time at I₁. In this case, the boundary intersects an adjacent pair of unresolved run-length records, the first one representing background pixels and the second one representing blob pixels. The update procedure under these conditions consists of simply replacing the -1's by the correct values. The IR field of the background run gets I₁-1, and the IL field of the blob run gets I₁. Figure 17 shows the evolution of a portion of the run-length list for a simple image.

The initial crossing of a run of pixels can also occur on the left, i.e., at I₁, instead of on the right as described above. If the boundary tracker is following an edge downward in the image (increasing J) and then the edge swings upward again (due to a concavity on the top edge of the blob), the pixel that runs on the rightmost projection of the boundary will be initialized on the left first. An example of this is shown at point A in Figure 18. Handling this case requires a set of procedures analogous to those described above, linking in run-length records that leave the right end undetermined initially and resolve it when the boundary recrosses the line on a downward segment.

Another situation can arise due to concavities in the boundary: the boundary may intersect a completed run of pixels that is already marked as part of the object being traced. This is essentially the same as the first case described, where a "down" segment intersects a background run, except now a piece of background has to be "cut out" of the object. The crossing can occur on the right with an "up" segment, as shown at point B in the figure (the completed object run would be from point C to D, or on the left with a "down" segment, as shown at E. In either case, a background run with one end unresolved is spliced into the object run, and later corrected at the second crossing of the line as before.

If the boundary tracker is invoked starting at a point inside a concavity of the boundary, then one other special case can occur. The boundary crosses between two unresolved runs, which would normally resolve them, but the crossing is in the "wrong" direction, i.e., the boundary segment orientation implies that the object is on the left, but the left run is a background run. By always starting the tracker on a convexity of the boundary, as in the case of the algorithm described here, which invokes the tracker on the first image line from the top that encounters the object, this problem cannot occur.

When the boundary tracking algorithm has finished processing the image, the resulting blob list will have all the hierarchical linkages between blobs in the scene, just like those produced by the connectivity analysis method (though the order of blob allocation, and thus the order of blobs in the various lists, may differ). In addition to the blob statistics, however, each blob record generated by the boundary tracker has a pointer to the chain-code list that gives an exact description of the boundary. The moments accumulated by both algorithms are sufficient for computing a blob's centroid and major axis (angle of minimum moment of inertia), but if higher-order moments are needed, the chain-code list can be traversed to derive them. Alternatively, of course, fields for additional statistics could be added to the blob record and their values accumulated during the first traversal of the boundary.

Regardless of which scene analysis algorithm you choose, the result you get is a complete description of the patterns of black and white in the image. I should point out that even if you are restricted to using connectivity analysis in a single raster scan of the image (due to the practical limitations of your image access method, for example), you can still get a chain-code description of the blob outlines if necessary by modifying the boundary tracking algorithm to accept the run-length list as input. If efficient random access to the image is available, however,

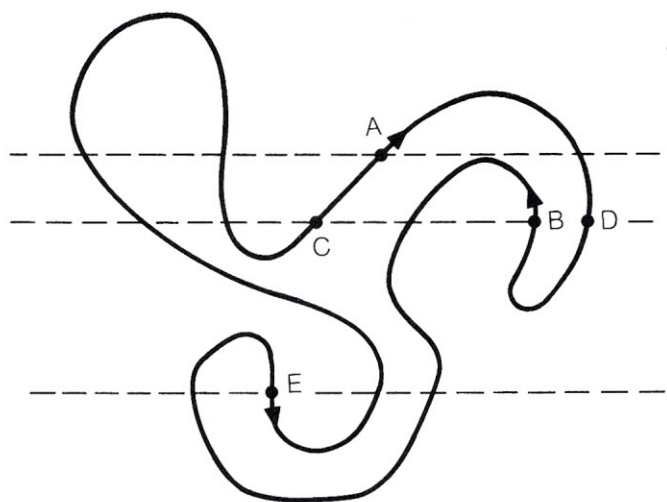


Figure 18. Concavities in the outline of a blob can give rise to cases that require special handling. At A, the initial crossing of a run occurs on the left. At B a background run must be "cut out" of run CD. A similar situation occurs at E.

it's better to use the boundary tracking algorithm first if you need chain-code.

Using the Image Description

Beyond simply locating targets in two dimensions, applications of robot vision usually involve three basic categories: object recognition and inspection, stereo vision for range measurement, and tracking moving targets. Within the particular limitations of binary images, the basic blob description approach described here can be used to advantage for each of these functions. Although a complete treatment of each of these topics will be reserved for future articles, there are a few comments pertaining to the use of blob descriptions that should be addressed here.

Object Recognition

In its simplest form, object recognition consists of matching a blob of unknown category against a finite set of possible objects. Each possible object is represented in a *model* in terms of features that can be derived from information stored in a blob record. There are two basic approaches to modelling, and hence to recognition. One is purely numerical. Objects are modelled by a list of n different features, such as area, or moment invariants, perimeter, each of which can be expressed by a single number. It is convenient to think of the values in the list as the coordinates of a point, or *feature vector*, in an n -dimensional feature space where each axis representing ideal descriptions of each object, perhaps with a small envelope of acceptable variations about each point, that is gained by a "training session" where the system is shown sample objects. A blob can then be recognized by finding its "nearest neighbor" among the set of model points in feature space, possibly with a minimum acceptable "degree of match." [refs. 1,7]

The other approach to modelling represents the shape of each object explicitly in terms of its boundary. This is usually done by approximating the chain-code description of the boundary as a sequence of straight line segments and possibly circular arcs. In this case, recognition is either based on template matching to find the boundary description in the model which most closely matches the boundary of a particular blob in the image, or else the boundary is used to verify or possibly disambiguate matches based on numerical features. [ref. 8]

Because the only information available is derived from the shape of the outer boundary of a blob, binary image vision using these techniques is particularly ill-equipped to recognize objects in three-dimensional scenes. The shape

of an object's outline, and hence its blob statistics, can change dramatically when viewed from different perspectives, even if it is possible to provide sufficient contrast between object and background by suitably structuring the scene. Matters are made worse by occlusion, the apparent juxtaposition of objects from a particular point of view. In this case, the two objects would be merged into one binary blob in the scene.

Stereo Range Measurement

The location of an object in 3-space cannot be determined from a single view of the object unless there is some known constraint on one dimension. [ref. 9] Usually, the constraint is that the object lies in a plane whose position relative to the camera is known. In the absence of such a constraint, we need at least two views of the object from different camera locations. These can be provided by a stereo pair of images taken by two cameras mounted some distance apart.

The problem in making stereo measurements is to match points from the two images which correspond to the same point in 3-space. One of the classical approaches is *stereo correlation*, which relies on a statistical correlation between the distribution of gray level values in a neighborhood surrounding the target point in the two images. [ref. 10] This method is not particularly suited to binary images, since the correlation is undefined unless there is some variation in gray level. Thus, the only possible basis for correlation would be the shape of the boundary in the neighborhood of an edge point—a very poor basis for discrimination if there is more than one object in the scene.

Fortunately, we can remove the ambiguity in matching edge points if we initially perform *global* blob matching. For any point in one image, the stereo camera geometry model specifies a line segment in the other image which contains the matching point [9]. Taking as a target point the centroid of a blob from one image, the centroid of the corresponding blob in the other image should lie near this line, unless the difference in perspective is too great (which can happen if the target is too close to the cameras, for example). Once we have identified candidate matches on the basis of camera geometry, then the blob statistics can be used to select the best match. You can then match edge points to provide several 3-D measurements of points on the border of the object. Knowing the 3-D location of three non-colinear known features of the object is sufficient to completely determine its position and orientation in 3-space.

Tracking moving targets

Provided that the target or some feature on it has sufficient contrast to be readily distinguishable from the background, binary vision using blob descriptors may be sufficient to measure its translational and rotational velocity in a succession of images. Here again, the centroid of the target blob is the primary source of information. Given the difficulties imposed by the need for high contrast and lack of occlusion, the major concern in the context of robotics is to execute a tracking algorithm in real time. If the results of object tracking are to be used as feedback in a control loop, as in visual servoing, for example, then the tracking should be continuous and there should be as little delay as possible from the time an image is formed to the time new target position and velocity measurements are available.

In most cases, real time tracking systems are designed to operate at 30Hz, corresponding to the 30 frame/sec. rate of a standard video signal. The first thing required to achieve this processing rate is a digitizer system that operates at the full video rate, such as the one mentioned earlier [3]. Secondly, the image analysis and tracking software is usually written in assembly/machine code optimized for speed. Thirdly, rather than attempting to analyze the entire image, the algorithms are modified (like BLOBSCAN in the listing) to operate within a window into the image which can be accurately placed where the target is expected to be in a given frame, based on its position, velocity and acceleration calculated from previous frames and allowing for uncertainty in the prediction. The higher the cycle rate of the algorithm, the less the appearance of the target is likely to change from one image to the next.

Conclusion

The techniques and applications I've described here should give you some idea of the potentials of robot vision using binary images—as well as an appreciation of its limitations. The algorithms also give you the basic tools for building your own binary image vision software. Under the right circumstances, these methods can give you a powerful tool for providing a robot with valuable sensory input at a very low cost in hardware and development effort.

References

- [1] G. J. Agin, "Computer Vision Systems for Industrial Inspection and Assembly," *Advances in Computer Technology*—1980, Volume 1 sponsored by ASME Century 2, San Francisco, Calif., August 12-15, 1980, pp. 1-7.
- [2] M. R. Ward, L. Rossol and S. W. Holland, "CONSIGHT: A Practical Vision-Based Robot Guidance System," 9th International Symposium on Industrial Robots, Society of Manufacturing Engineers and Robot Institute of America, Washington, DC, March 1979, pp. 195-212.
- [3] R. Eskenazi, "Video Signal Input," **Robotics Age**, Vol. 3, No. 2, March/April, 1981.
- [4] J. S. Weseka, "A Survey of Threshold Selection Techniques," **Computer Graphics and Image Processing** 5, 1976, pp. 382-399.
- [5] J. M. Wilf, "Chain Code," **Robotics Age**, Vol. 3, No. 2, March/April, 1981.
- [6] D. L. Milgram, "Constructing Trees for Region Description," **Computer Graphics and Image Processing**, Vol. 11, 88-89 (1979), 88-99.
- [7] R. Wong and E. Hall, "Scene Matching with Invariant Moments," **Computer Graphics and Image Processing**, Vol. 8, 1978, pp. 16-24.
- [8] W. A. Perkins, "A Model-Based Vision System for Industrial Parts," **IEEE Transactions Computers**, Vol. C-27, 1978, pp. 126-143.
- [9] A. Thompson, "Camera Geometry," **Robotics Age**, Vol. 3, No. 2, March/April, 1981.
- [10] Y. Yakimovsky and R. Cunningham, "A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras," **Computer Graphics and Image Processing**, Vol. 7, 1978, pp. 195-210.

by
Vern Mangold
Kohol Systems
P. O. Box 1185
Dayton, Ohio 45401

THE INDUSTRIAL ROBOT AS TRANSFER DEVICE

Since nearly every U.S. industry demands more productivity, industrial robots are proliferating. However, as robots make their way into our factories, we find that most of them are only used in *tool handling* applications—spot welding or spray painting, for example. Yet there exists another class of applications, one that has a tremendous untapped potential. This class uses *work piece handling* robots, which transfer products from one work station to another. Although tool handling tasks, such as spot welding, are readily performed by robots, I believe we should also consider applications that involve the transfer of products within the plant.

The concept of transferring a product from one work station to another is relatively new. Centuries ago, products were manufactured in roughly the same location. A single craftsman or mechanic simply changed tools when he started a new manufacturing task. The modern assembly line changed that arrangement. With the assembly line, factories had to transport products to dedicated areas, each of which was assigned to one specific subtask.

In many cases, robots can dramatically reduce the need to invest in parts transfer equipment designed for one particular part or process. Fixed automation equipment can be supplanted by a combination of programmable robots and *generalized* parts transport equipment—conveyors or pallets, for instance. Since you can often

produce different product styles by simply reprogramming the same robot, the same manufacturing line can produce different product batches. This brings the benefits of automation to factories whose production volumes would not justify customized “hard” automation.

How is Part Transfer Done Today?

Though industry uses many types of transfer devices, we can group these into two main categories: *bulk transfer devices*, which transfer randomly placed parts, and *orientation transfer devices*, which hold the workpiece in position. Of the two, industry more frequently uses bulk transfer devices. Conveyor belts, sheet metal chutes or guides, and simple part slide mechanisms are all devices of this kind. The familiar tote bin—often assuming different names, such as gond, wire basket, or part box—is the most basic form of bulk transfer device.

If you use machines that can only be fed with parts precisely positioned, then you need an orientation transfer device. The actual device you choose typically depends on how accurately you need to locate parts.

The overhead conveyor is a simple type of orientation transfer device which you can configure in one of three ways: indexing from position to position, continuously

moving, or *powered and free*. In a powered and free conveyor system—the most sophisticated of the three types—hanging hooks or racks move continuously around a closed loop. When a work station needs a hook at a given point, it can divert the hook from the main conveyor system and rigidly lock it into position. After the hook is loaded with a product, it returns to the main loop, from which it can move to a different work station. In this type of system, as well as in other overhead chain conveyor systems, gravity orients the part.

Another type of orientation transfer device is the *pallet conveyor*. This device is simply a normal conveyor equipped with pallets for individual parts. Parts are placed into the fixture of a part pallet and then transferred to another work station, either by continuous motion or by indexing in controlled steps. This type of transfer device can be extremely accurate. By using shot pins and other mechanical devices, a robot can locate a palletized part within 5 mils (.005").

A *lift and carry device*—which uses a mechanical linkage system to index parts from one work station to the next—is similar to the pallet conveyor. Like the pallet conveyor, it has high accuracy and good part repeatability.

Other types of orientation transfer devices include the *bulk hopper feeder with orienter* and *iron man* type devices. The bulk hopper feeder lets you automatically feed loose, randomly oriented parts from a bowl-like vessel by elevating individual parts to a drop-off position. Using the right fixture at the drop-off point, you can accurately deliver the part to a conveying device. An iron man operates by closing around a part and orienting it by the pressure of its grasping surfaces.

The question naturally arises: How much do these devices cost? Table 1 compares the costs of some of the transfer devices you might use to move a part about twenty feet, with accurate positioning in some cases. (Note that the prices given represent approximate data, gathered from industrial suppliers in the midwest region of the U.S. Prices, of course, will vary with geographical area and industrial activity.)

When evaluating the cost effectiveness of a transfer device, remember that you need a significant amount of manual labor when you use the most "economical" methods listed in Table 1. Consider the typical use of a tote bin, for example. After a work station is finished with a part, a worker manually loads the part into a bin for transport. When the tote bin arrives at its destination—usually by fork lift truck—another human worker unloads it. Transport by tote bin is labor intensive.

With the rising cost of labor, the relatively inexpensive transfer devices—such as tote bins, baskets, parts pallets,

and so on—become less attractive. In the U.S. automotive industry, for example, the average first shift worker costs around \$30,000 a year (if we include benefits and overhead). Clearly, we need ways to transfer parts with as little human intervention as possible.

Why a Robot?

Strangely enough, there has been no stampede of managers eager to use robots as transfer devices. Yet robotic technology has been proven in this country since 1961. Industrial robots have given millions of hours of productive service with availability (uptime) records of better than 98%. Industrial robots are physically able to transfer a wide variety of parts. Computerized servo-controlled robots can produce a broad range of complex motions in space and, with few exceptions, can be easily equipped with pneumatic clamps for handling parts.

Before we consider the advantages of using robots to transfer parts, we should examine the manufacturing process in more detail. In general, a finished product is the result of a number of sequential operations: fabricating, machining, assembling, testing, coding, packaging, and shipping, for example. But regardless of the number and type of operations a product requires, there are really only two basic ways to configure station-to-station parts flow: for *serial* or for *parallel* manufacturing operations.

The goal, when you design a manufacturing process, is to combine operations in a way that optimizes the

TABLE 1

Item 1	Tote bin—\$125 to \$250 each
Item 2	Gravity roller conveyor—\$2,200 to \$4,000
Item 3	Power roller conveyor—\$6,500 to \$9,000
Item 4	Gravity roller conveyor with fixture and escapement—\$7,200 to \$11,000
Item 5	Power roller conveyor with fixture and escapement—\$14,000 to \$35,000
Item 6	Part pallet—\$100 to \$175
Item 7	Part pallet with donnage—(ten layers) \$400 to \$550 each
Item 8	Indexing chain conveyor—\$12,500 to \$14,000
Item 9	Continuous chain conveyor—\$9,000 to \$11,000
Item 10	Powered and free accumulating conveyor—12,000 to \$16,000
Item 11	Indexing pallet conveyor—\$25,000 and up
Item 12	Iron man carry system—\$20,000 and up
Item 13	Lift and carry device—\$30,000 and up

Note: all prices include internal installation cost.

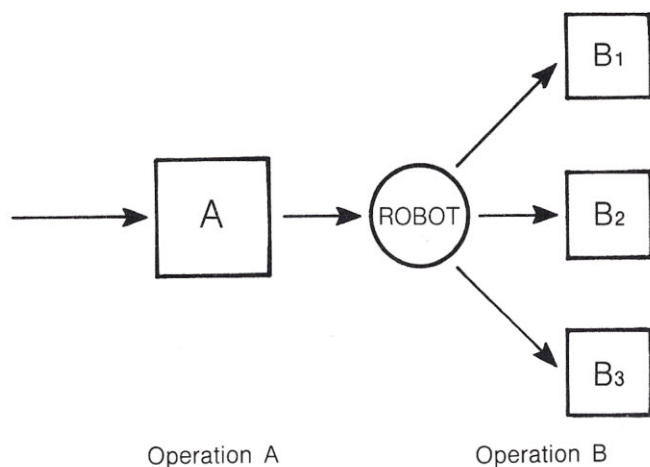


Figure 1. Serial and parallel processes are combined to achieve maximum throughput. The combination depends on the cycle times of each operation. In this case, operation B has three times the cycle time of operation A.

throughput of the process. To do this, you often have to process a product in parallel through an operation that has a relatively long cycle time—using several identical machines to perform the same function. For operations with relatively short cycle times, you may be able to use a single machine and still balance the process flow. This is serial operation. The processing mode you choose depends entirely on the individual cycle times of the operations involved.

The industrial robot can be effective in both serial and parallel operations. In a serial operation, momentary imbalances in the cycle times of two or more operations will interrupt smooth product flow. You can use the flexibility of an industrial robot to compensate for this by programming the robot to offload the product into a buffer storage area. In this way, even when unequal cycle times and the downtimes of capital equipment interrupt the flow of product through a process, manufacturing operations do not have to come to a standstill.

In a parallel operation, you can program an industrial robot to service on demand two or more identical devices. You can maintain your throughput on operations with longer cycle times by letting the robot load processing elements as they become available.

For the purpose of discussion, consider a hypothetical part transfer task. Suppose the part must be transferred from a prior operation to a process device called A. After process operation A has been completed, the part must be transferred to operation B. Assume that operation B has a cycle time that is three times longer than operation A. To maintain a constant throughput, three identical work stations, each performing operation B, will be clustered with the work station for operation A. After completing operation A, we transfer the part to operation B1, B2, or B3. Figure 1 diagrams this process. In this way, we can process the parts through a work area with maximum efficiency and balanced flow.

In both the serial and the parallel cases, industrial robots have an advantage over dedicated transfer devices. Industrial robots are not limited to one fixed mode of operation. You can program them to perform both serial and parallel operations in one work area. Robots can place parts much more accurately than simple conveying mechanisms: there are commercially available industrial robots whose repeatability is within 4 mils. By judiciously selecting a robot and effectively designing its tooling, you can get performance accurate enough to do nearly any kind of parts transfer.

Industrial robots are extremely reliable. No robot could survive in an operation that has a greater uptime than the robot itself. Manufacturing managers cannot tolerate any automation device that becomes, through its own downtime, the limiting factor in the operation. For this reason, quality control in manufacturing the robots themselves has become perhaps the most stringent of any industry. This strict control also helps reduce operating costs, since it lessens the robot's need for maintenance.

The majority of industrial robots in the U.S. can boast of an operating cost of less than five dollars an hour. This figure includes the base price of the equipment and is calculated over a two-shift, twelve month year. Since first shift labor costs average about seventeen dollars an hour, the industrial robot can easily cost less in the long run than manual transfer.

Robots also save labor costs by eliminating the equipment needed to protect the safety of human workers. Using industrial robots for parts handling can sizably reduce costs in shielding, guarding, ventilating and lighting. It also reduces the insurance costs and liability risks associated with human labor.

When you calculate how much manpower is replaced with a robot, remember that since a robot works more than one shift, it can replace more than one worker. It can also replace more than one man per shift—because two or more workers are often needed to move a heavy workpiece. In addition, a normal production worker will grow tired during the course of a typical work shift. It is a general rule in industry that, in 95% of the applications, industrial robots take *more* time than manual labor to perform a given task. However, averaged over the course of an entire work shift—in nearly every case reported—robots produced a higher *total* number of finished parts. This is entirely due to fatigue and “downtime” for human needs. It is not uncommon for managers to justify a robot installation solely on the direct replacement of first shift manpower.

In many cases, parts handling critically affects the quality of a product. Consider an extreme example—steel

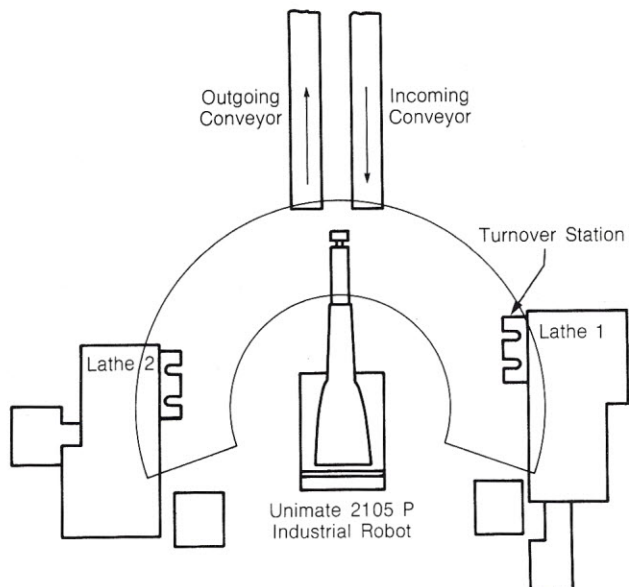


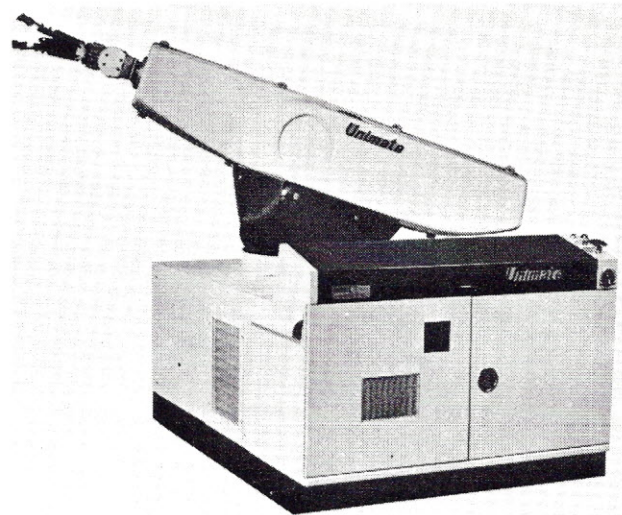
Figure 2. A Unimate 2000 Series robot transfers steel forgings in a serial process.

forgings. A billet of steel must be kept within a limited range of temperatures. Therefore, any delay in handling the product will cause incorrect forging and, in some cases, extensively damage the dies. With steel billets, the consistency of the transfer process is as critical as the consistency of the steel and the force of the forging press. A robot can insure that consistency.

O.K. So How Can I Use My Robot?

Rather than discuss the "general case," let's look at some examples in which robots are used as transfer devices. The following five cases offer a good cross-section of this kind of application:

Serial Operation is shown in Figure 2. The problem is to transfer steel forgings through two successive machining operations. The parts are six kinds of forged steel housings that weigh from 2 to 40 pounds each. The outer diameters of the parts are about 14", and their thicknesses are about 3". This application uses a Unimate 2000 Series robot. The Unimate accepts raw parts from an incoming conveyor and preloads the part on a positioning table in front of a Jones and Lamson (J&L) turning machine. When the J&L machine has completed its cycle, it signals the Unimate. The robot extracts the semi-finished part, places it on a second rest stand, and loads another raw part into the first J&L. When the raw part is correctly positioned in the first J&L, the robot retracts and signals the machine to begin its cycle. The robot then transfers the semi-finished part to a third rest stand for preload into the second J&L. Upon receiving a signal that the second J&L has completed its cycle, the robot extracts a finished part and places it on a fourth rest station. As in the first machine operation, the



Unimation's UNIMATE Robot—widely used for parts transfer applications.

robot loads the semi-finished part into the second J&L, retracts, initiates the machining cycle, and deposits the finished part on an outgoing conveyor. The robot costs \$80,000, and its payback period is about 9 to 11 months of two shift operation.

VENTURE CAPITAL

Idanta Partners is a venture capital firm that invests in and provides support for new and growing companies. We invest at various stages of corporate development, but specialize in assisting able and experienced managers with the formation of new businesses that we finance.

We would be interested in assisting executives of demonstrated competence with the organization and development of a major company in the robotics field.

Brochure available on request.

IDANTA PARTNERS

3344 North Torrey Pines Court, Suite 200
La Jolla, California 92037

CIRCLE 19

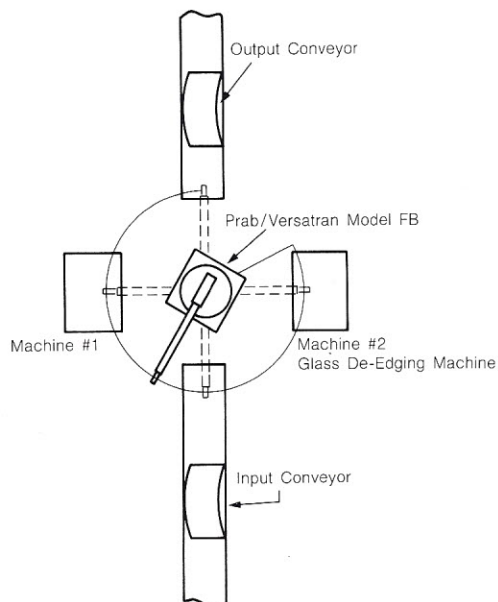
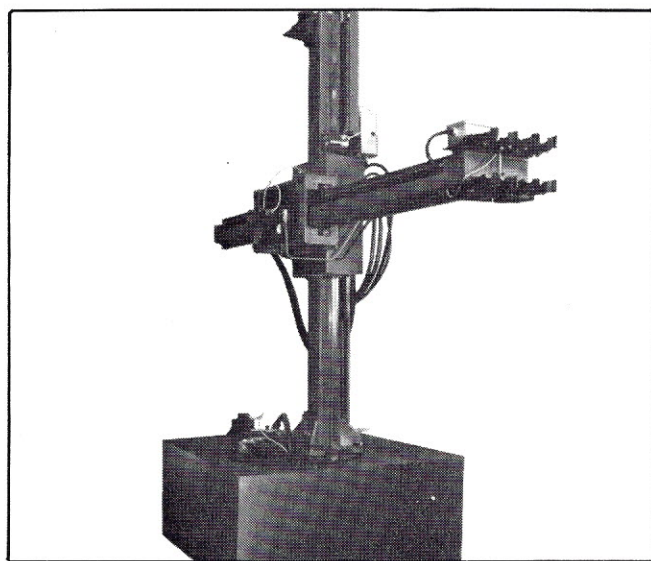


Figure 3. A Versatran FA Series robot is used here to transfer automobile windshield and window glass in a parallel process.

Parallel Configuration is shown in Figure 3. The task is to transfer glass windshield and window panels from a washing conveyor to a glass de-edging station. The panels weigh from 6 to 48 pounds and range in size from 6" by 4" to 54" by 22", each .25" thick. The tooling on the glass de-edging station has three stops that must be positioned within 5 mils. In this application, a Versatran FA Series robot with 660 control transfers the panels from the washer conveyor to the de-edging station on demand. The robot selects the appropriate de-edging machine based on input signals from either machine. To optimize cycle time, the robot is equipped with a dual hand. The robot's hand tooling is adjustable to handle the full range of the 22 different parts required in this application. The transfer's

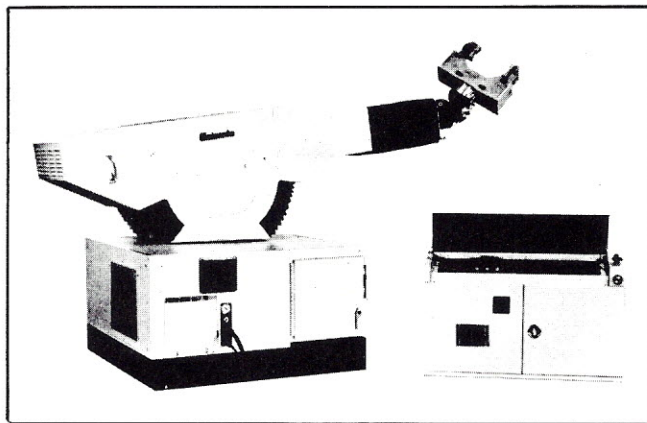


Versatran Model FA Robot

cycle time is 5.5 seconds. The Versatran costs about \$65,000, and its payback takes around 13 to 14 months of two shift operation.

A process that combines serial and parallel operation is shown in Figure 4. Steering knuckles for large off-the-road earth-moving equipment must be processed through a numerically controlled (NC) turning machine, a gauge station, and a dedicated transfer line for drilling and counter-boring. Because of the NC machine's cycle time, the factory needs three of them to maintain throughput. The steering knuckles weigh 78 pounds, are 7" high, 8" wide, and 3" thick. A Series 4000 Unimate robot moves the steering knuckles through available NC turning machines, then takes the finished parts through the gauge and subsequent transfer machining operations. The Unimate is equipped with a dual hand to optimize cycle time. Ten seconds before the completion of a machine cycle on the NC turning machine, it sends a pre-initiate signal. This allows the robot to position itself in a "ready" position just outside the NC machine. Cycle time for this operation is 8 seconds for each part transfer to the turning machine and 6 seconds for the gauge station cycle. The Unimate costs \$120,000, and its payback takes about 15 to 18 months.

Random transfer is shown in Figure 5. Various appliance components require transfer from one moving conveyor to another. In this case, the parts are transferred from an unpalletized part conveyor to a conveyor that transports the parts to a dipping tank. After the parts have been dipped, they must go back to the original conveyor. The parts are appliance components, weighing from 10 to 15 pounds and coming in assorted sizes. A Cincinnati T3 robot, with a tracking option, performs the transfer. It is



Unimate 4000 Series

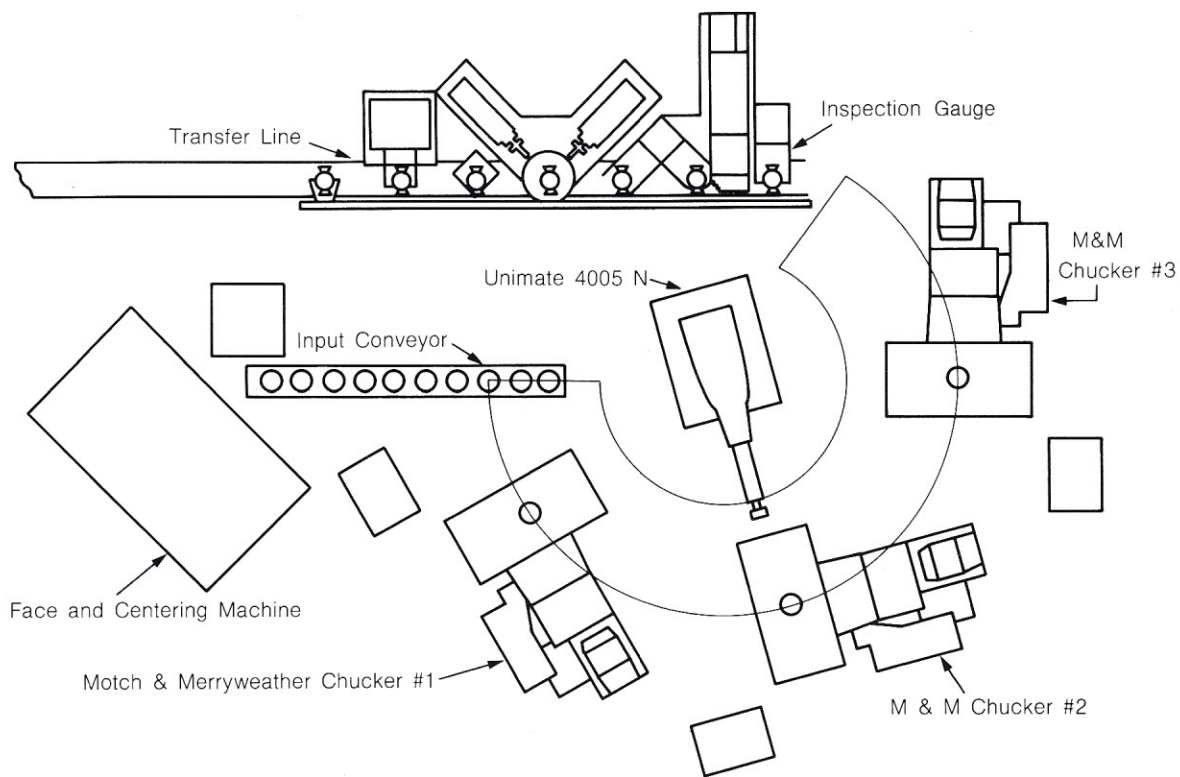


Figure 4. To transfer the steering knuckles of large earth-moving equipment, a Series 4000 Uimate robot is configured to combine serial and parallel process.

equipped with special tooling that allows it to pick up a wide variety of parts. Both conveyors are continuously moving—at different rates. Steel hooks carry the parts on the conveyor. The robot has a cycle time of from 4 to 7 seconds to transfer parts from conveyor to conveyor, depending on the part. The Cincinnati costs \$96,000, and its payback takes approximately 10 to 11 months of three shift operation.

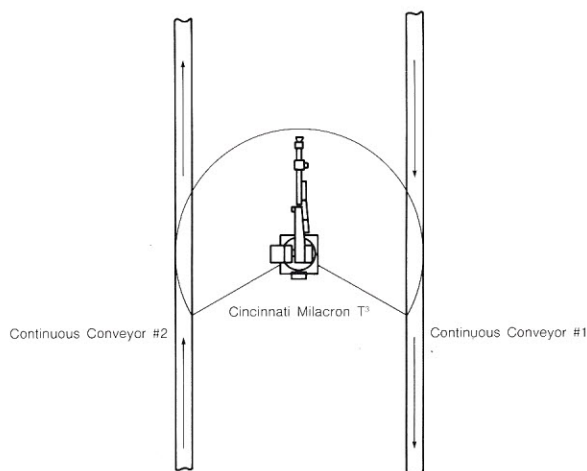
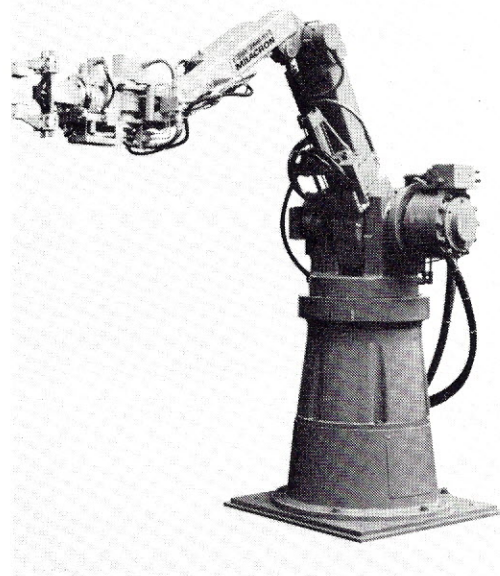


Figure 5. A Cincinnati Milacron T³ robot transfers appliance components in a random select process.



Cincinnati Milacron's T³—able to lift a hundred pound part.

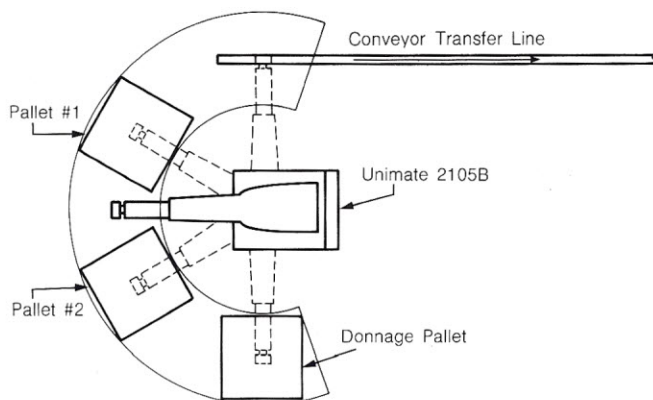


Figure 6. A Unimate 2000 Series robot unloads engine heads from parts pallets.

Palletizing is shown in Figure 6. Broached engine heads for four cylinder engines need to be loaded into pallets then transferred to another plant for machining. The parts must be unloaded from the pallets onto a conveyor that feeds a transfer machine. The engine head weighs 42 pounds, and has the dimensions of 18" high by 4" wide by .75" thick. In this case, a 2000 Series Unimate robot works in conjunction with three pallet locaters. The locaters are fixed to the floor adjacent to the robot. Pallets loaded with 96 parts (6 layers with 16 parts per layer) are placed into the locaters for accurate positioning. The robot automatically selects the proper program for de-palletizing a given pallet. When the robot finishes a layer of parts, it removes the donnage* separators and deposits them in another pallet location. The tooling is designed to indicate whether a part has been properly loaded into the donnage. If a part has been loaded incorrectly, the robot hand tooling will signal the robot to select a different program in order to properly orient the part. After the robot has finished unloading one pallet, it will automatically select a program to unload the second pallet while giving a signal to an outside operator, who places a full pallet of parts in the empty parts locater. The entire engine manufacturing cycle time is 18 seconds, of which the robot takes 6 seconds. Since the whole plant is paced on an 18 second cycle, the robot has a larger percentage of dead time in this application. Still, the robot costs \$45,000, and its payback period is only about 5 or 6 months of two shift operation.

Conclusion

Although I've spoken favorably about using robots as transfer devices, I should point out that robots do have some limitations. Currently, robots have very limited

*Donnage is a device, often a plastic mold, placed on top of a pallet to precisely locate the palletized object for a robot.—ed.

TABLE 2

Robot Manufacturer	Type	Approximate Weight Carrying Capacity	Price Range
ASEA	DC electric servo, microprocessor	15 to 150 lbs.	\$96,000 to \$125,000
Auto-Place	Non servo pneumatic, iterative processor	20 lbs.	\$4,000 to \$11,000
Cincinnati Milacron	Hydraulic servo, microprocessor	100 to 250 lbs.	\$67,000 to \$125,000
Prab	Non servo hydraulic, iterative processor	50 to 125 lbs.	\$25,000 to \$35,000
Prab-Versatran	Hydraulic servo, microprocessor control	80 to 2,000 lbs.	\$45,000 to \$95,000
Seiko	Non servo pneumatic, iterative process	1 to 10 lbs.	\$6,000 to \$9,000
Unimation	Hydraulic servo, microprocessor	50 to 450 lbs.	\$27,000 to \$60,000
Unimation-PUMA	DC electric servo, microprocessor	3 to 7 lbs.	\$37,000 to \$42,000

Note: Actual weight carrying capacity figures are estimates only. The load carrying capacity of any industrial robot is dependent upon part configuration factors and application parameters.

sensory perception. They have to rely on relatively crude sensing techniques to orient and align parts. This means that you have to engineer a significant amount of structure into the manufacturing environment before a robot can function there successfully. Today's robots can't just reach into a tote bin and pull out the right part!

Giving the robot a properly oriented and accurately located part can be expensive. Methods as economical as a bulk flow feeder are often as expensive as the robot itself. This lengthens the robot's payback period.

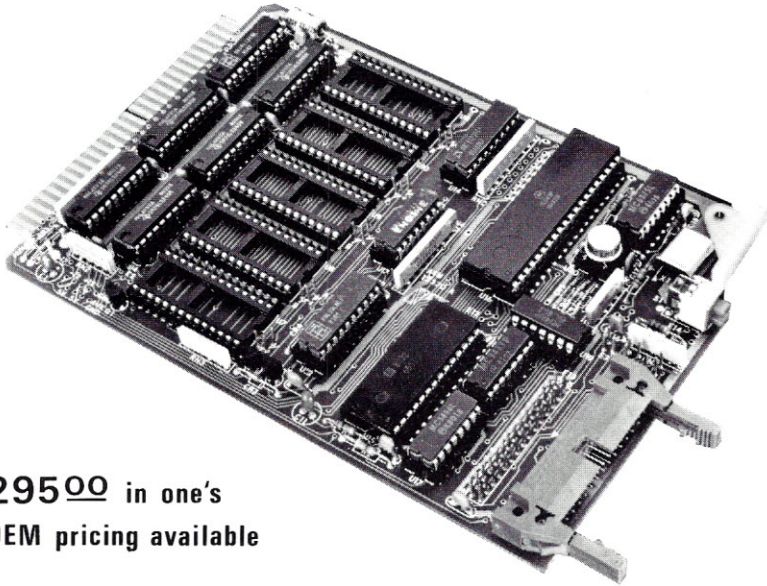
The robot itself, especially for the first-time user, is not inexpensive. Table 2 gives the prices and capabilities of some industrial robots. Added to the base price of a standard robot, is the cost of hand tooling and accessories. A hand can cost from \$4,000 to \$25,000, depending on the application. The price of the engineering needed to expedite a project can run from \$4,500 to \$15,000, depending on the application. Add to this the cost of one-time accessory items—programming tools, program storage options, and spare parts kits, for example. All these factors inflate the bottom line cost of a robotic project.

Still, I believe that none of the disadvantages mentioned are significant enough to stop the widespread use of robots in transfer applications. The major restraint, at this time, is the lack of education on the part of plant, manufacturing, and industrial engineers and management. As these groups learn how to effectively use robots, the robot as transfer device will come into its own.

STD BUS '6809'

The newest in Datricon's family of low-cost 4th engines.

MEMORY capacity to 40k bytes, **RS232/RS422 SERIAL PORT** and the powerful **6809 CPU**, all on one 4.5 x 6.5 inch **STD Bus Card**



295⁰⁰ in one's
OEM pricing available

These Specifications Tell You More!

Every way you look at it, this powerful STD Bus Processor uses Industry-wide Standards.

STD Bus interface (both STD-Z80 and STD-6800 compatible) offers unprecedented user support with Analog, Power Input/Output, Disk and advanced communications protocols.

SERIAL PORT supports RS232C or RS422 with full modem controls including software baud rate, from 50 to 19.2Kbaud. User selectable standard since RCVR/DRVR's are factory installed.

BYTE-WIDE MEMORY concept permits the use of 20 currently available memory devices from 2k x 8, 4k x 8, and 8k x 8 RAM, ROM and EPROM.

QUALITY AND RELIABILITY

Backed by Datricon's standard one year parts and labor warranty, 200 hour burn-in and extensive factory testing, our customers are assured of receiving high quality product.

D-FORTH SOFTWARE

Datricon's popular D-FORTH software available on the Series 12 and 14 is also available on the Series 09. Optimized for control systems, D-FORTH is high-level and interactive, it is especially useful in interactive control applications such as testing and process monitor/control. Efficient memory utilization and rapid execution provide exceptional Return On Investment.

Contact Datricon's nationwide staff of highly qualified sales representatives or the factory for information.



QUALITY WITHOUT COMPROMISE

503 - 284-8277 7911 NE 33RD Drive Portland, Or 97211

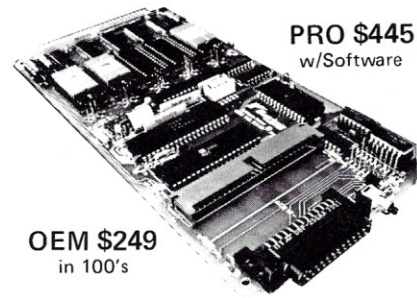


By
Datricon Corporation

'THE SINGLE BOARD SYSTEM'

- Two 16 bit timers
- 16 bit parallel I/O
- RS232C, (programmable)
- Up to 40K bytes memory
- STD Bus compatible

(Needs only power supply and terminal to operate)



PRO \$445
w/Software

OEM \$249
in 100's

ROM RESIDENT D-FORTH

- Generates ROMmable Applications Code
- Development System Available (ADS120)
- Functionally Compatible with most FIG-Forth

UP TO 40K BYTES OF MEMORY

- BYTEWYDE(TM Mostek) concept allows use of 16k, 32k, 64k and 128k RAMS and ROMS

STD Bus

- STD Bus Protocol Compatible with 6800 and most Z80/8080 Peripherals
- STD Bus Support. Backplanes, cages, extender boards (EXT 12) and STD Bus Diagnostic Tool (DDT 12)

PROCESS CONTROL, SMALL OR LARGE

- 6800 micro processor driving 16 I/O lines, two 16 bit timers; software programmable serial port (including baud rate) with full modem control on RS 232C

APPLICATIONS

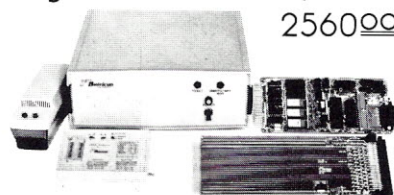
- Direct interface to popular 8 or 16 channel AC/DC I/O industrial panels
- Direct interface to P521 and P611 Periphicon optical image digitizers

RELIABILITY

- 200 Hour Burn In
- One year warranty on all workmanship and parts

High Level Development

2560⁰⁰



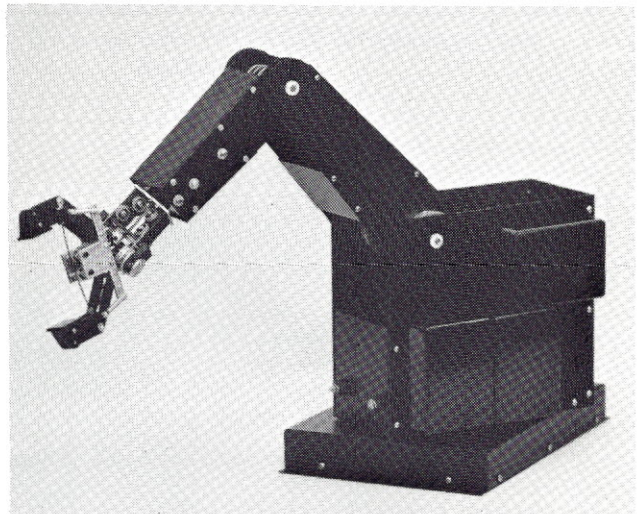
QUALITY WITHOUT COMPROMISE
503 • 284-8277

7911 NE 33rd Drive Port. Ore. 97211

CIRCLE 20

CONTINUOUS PATH CONTROL WITH STEPPING MOTORS

by
Shafi Motiwalla
General Electric Corporation
Schenectady, New York



Part One: Designing a Preview Controller

The capabilities of today's industrial robots vary over a broad range, from simple *pick and place* machines to sophisticated multi-processor controlled systems. Yet all industrial robots consist of two major components, the manipulator and the controller. The moving parts in the manipulator, which make up the arm, wrist and end-effector (gripper, etc.), must all be driven by some kind of actuator. At present, pneumatic and hydraulic actuators are the most commonly used. Hydraulic drives have a large lift capacity and offer good control in a compact package, but are expensive, have limited speed, and may pollute their workspace with hydraulic fluid. Pneumatics are relatively inexpensive, but the compressibility of the air makes them suitable only for movement between fixed mechanical stops. For the newer robots designed for assembly and other precision work, electric drive is becoming increasingly popular. The use of servo motors or stepping motors can provide fast, accurate control of the manipulator's position and velocity throughout its range of motion.

Apart from the physical limitations of the manipulator, the design of the robot's controller determines the capabilities of the system and its range of applications. The most common controllers can position the hand very accurately, but may offer little control of the robot's path and speed between target points. This latter feature, called *continuous path* control, is essential for many manufacturing operations that require the robot to follow a line or curve in space, as in the case of arc welding.

In this article I will describe a method of attaining continuous path control of a robot driven by stepping motors. This work was done in the Mechanical Engineering Department at the University of California at Berkeley. After first discussing the operation and advantages of using stepping motors, I will present a control scheme that uses "look ahead" to anticipate changes in path direction. This approach, using "preview" or "feed-forward" control, demonstrates the performance possible with stepping motors in an area thought to be restricted to expensive servo motor systems.

Stepping Motors as Actuators

Because it is possible to design very simple control systems for stepping motors, they are being used in a wide variety of applications. The idea of using stepping motors in control systems was proposed in the 1930's, where they were employed by the British Navy as remote positioning devices. [1, 2] Today the diverse applications of steppers include machine tools, process control, computer peripheral equipment, and recently, robots. In general terms, a stepping motor is an electromagnetic incremental actuator which converts digital pulse inputs to discrete motion steps. (See sidebar: "Stepping Motor Design.") In the familiar rotary stepper, the shaft rotates in equal increments in response to an appropriate input pulse train. There are a variety of stepping motor designs, the most popular of which is the permanent magnet rotary type. For a discussion of the capabilities and limitations of each type, see [3].

A stepping motor has several operating modes, depending upon the stepping rate and load conditions under which it operates. If the stepping motor is pulsed at a sufficiently slow rate, it comes to rest at the end of each step, moving in discrete steps as shown in Figure 1. In applications where the distance to be traveled is small and the response time is not critical, this mode of operation provides the simplest solution.

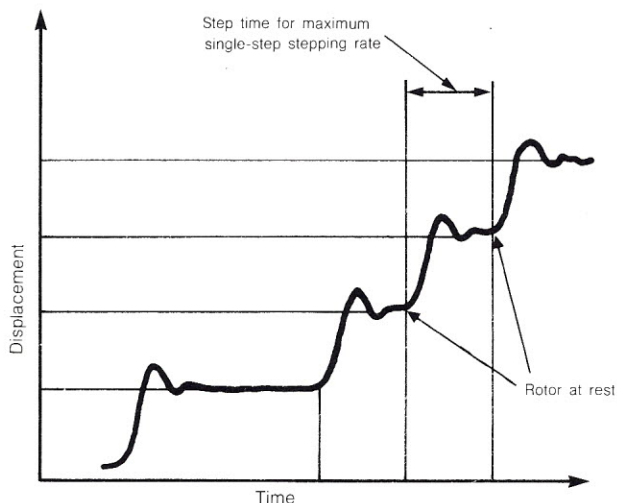


Figure 1. Displacement vs. time for single-step operation. For each step, the inertia of the motor and its load result in the step response shown, which has a momentary oscillation at the end of the step.

As the input pulse rate to the motor is increased, the motion changes from discrete steps to a continuous motion referred to as *slewing*. In the transition between these operating modes, the motor does not come to rest between steps, thus subsequent switching pulses may come when the motor has either a positive or negative velocity. Unless the pulse rate is chosen carefully, this can cause the motor to behave in a somewhat unpredictable and erratic manner. As shown in Figure 2, the velocity of the stepping motor in its steady-state slewing rate generally goes through oscillations about an average value.

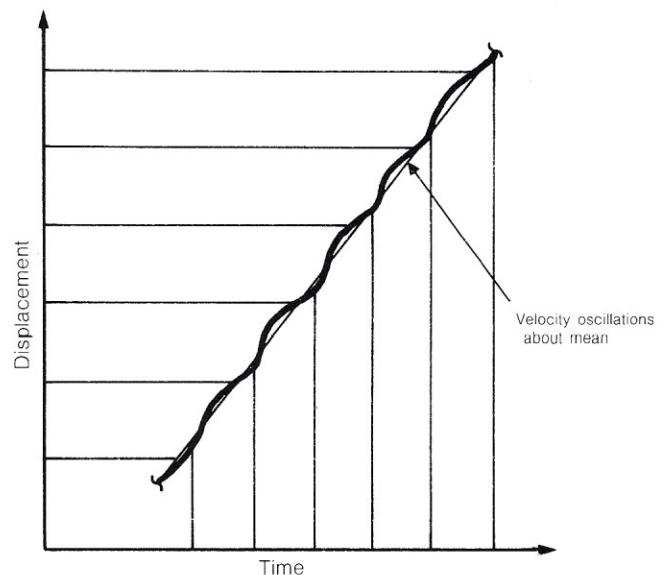


Figure 2. Displacement vs. time for steady-state slewing.

Figure 3 shows the torque vs. step rate curves for a typical stepper in each operating mode. Below the dotted curve for the discrete stepping mode, the load on the motor is low enough for it to step at the given rate without "missing" any pulses and with the motor returning to rest after each step. Operating in the slewing mode, however, the motor can produce greater torque at a given step rate, but changes to the step rate must be made carefully so that the peak torque is not exceeded. Above the peak torque, the motor may fail to step in response to an input pulse. One of the key advantages of using a stepping motor is that, by observing its torque/step rate limitations, you can use the motor "open loop" with full confidence that you can tell where its shaft is (relative to some initial position) merely by keeping track of the number of steps sent to the motor.

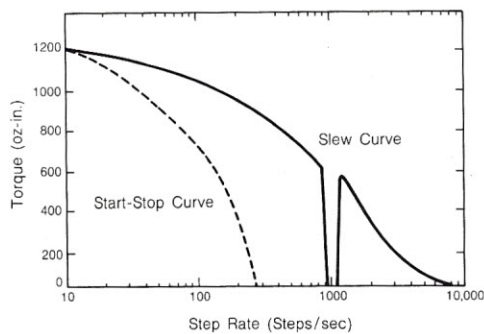


Figure 3. Speed-torque curves of a permanent magnet stepping motor. The notch in the slew mode curve is due to the mechanical resonance of the motor, which in this case occurs at a 1KHz step rate.

Control of Stepping Motors

The block diagram of Figure 4 displays the basic elements of a stepping motor control system. These consist of sequencing logic, power drive circuitry, power supply, current or voltage limiting circuitry and the motor. Position/velocity feedback may be used if necessary. The sequencing logic accepts the input pulses with the direction command and supplies low level signals to each of the power drivers. The power driver circuits take this low level signal and control the sequence of high currents from the power supply. A current and voltage limiting circuit is provided to buffer the power delivered to the motor phases to provide the discrete motion. [A working stepper motor control circuit is described in the May/June '81 issue of *Robotics Age*: "A Homebuilt Computer Controlled Lathe."]

Figure 5 illustrates the open-loop use of a stepping motor in a numerically controlled machine tool. Stepping motors are natural for this type of application since if properly controlled the positional error does not accumulate and no feedback is necessary. An example of feedback control using a stepping motor is shown in Figure 6. Although steppers are typically used open loop, feedback may be necessary for some applications, for example, when the load is unpredictable (which could cause missed steps), or when there is an inexact relationship between the movement of the motor shaft and the position of the load (due to gear backlash, for instance).

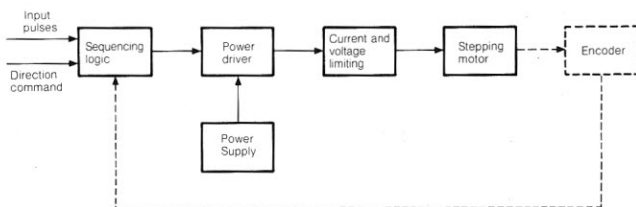


Figure 4. Basic elements of a typical stepping motor control system. The use of feedback can usually be avoided.

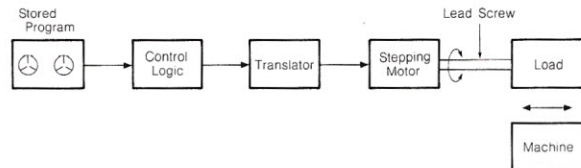


Figure 5. Open loop control of a machine tool.

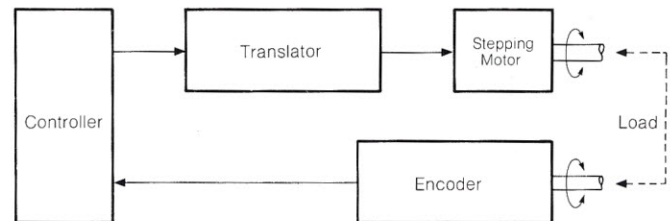


Figure 6. Closed loop control.

Although control systems using conventional a.c. or d.c. motors and prime movers can be designed to control incremental motion satisfactorily, a system with stepping motors offers the following advantages: (a) A stepping motor is inherently a discrete motion device. You can more easily interface it to other digital components; (b) The position error in a properly controlled stepping motor is non-cumulative, in that you can achieve accurate position and speed control with a stepping motor in an open-loop system. Transducers such as tachometers or position encoders can usually be eliminated; (3) Stepping motors can be induced to produce high holding torque during quiescent periods. This can eliminate the need for brakes or clutches; (d) If you stay below the single-step load curve, it is possible to start and stop stepping motors instantly; (e) Since a stepper uses digital power pulses, expensive linear power amplifiers are eliminated; (f) The design procedure for a stepping motor controller is normally simpler than for a servo motor controller. Recently, manufacturers have been putting increased effort on the development of improved stepping motors of smaller size and higher power, speed, accuracy and resolution. This has diversified the application of stepping motors to include those areas formerly limited to d.c. servo motors.

The Continuous Path Control Problem

Continuous path control techniques can be divided into three basic categories, based on how much information about the path is used in the motor control calculations. These are illustrated in Figure 7. The first is the conventional or *servo control* approach. This method uses no information about where the path goes in the future. The controller may have a stored representation of the path it is to follow, but for determining the drive signals to the robot's motors, all calculations are based on the past and present path tracking error. This is the control design used in most of today's industrial robots and process control systems. (An introduction to basic servo control has already appeared in *Robotics Age*. [4])

The second approach is called *preview control*, also known as "feed-forward" control, since it uses some knowledge about how the path changes immediately ahead of the robot's current location—in addition to the past and present tracking error used by the servo controller. A simple example of preview control is the case of driving a car. You don't wait until you're starting to drive off the road before starting to turn the wheel. Instead, by noting the curvature and grade of the road immediately ahead, you prepare for the change and initiate an appropriate action before you start to go astray. The controller I'll be describing here works somewhat like this, but, of course, in a much more restricted domain.

The last category of path control is the "path planning" or "trajectory calculation" approach. Here the controller has available a complete description of the path the manipulator should follow from one point to another. Using a mathematical-physical "model" of the arm and its load, it precomputes an acceleration profile for every joint, predicting the nominal motor signals that should cause the arm to follow the desired path. This approach has been used in some advanced research robots to achieve highly accurate coordinated movements at a high speed. [5]

A Preview Controller for Stepping Motors

In traditional control theory, variables are usually regarded as continuous analog quantities, and feedback loop designs also treat time as continuous. Analog control systems can be close approximations to these designs. When designing control systems that use digital logic, including computers, we must quantize time and allow for the time it takes to perform the control calculations. If the control loop is "running" at a constant cycle time,

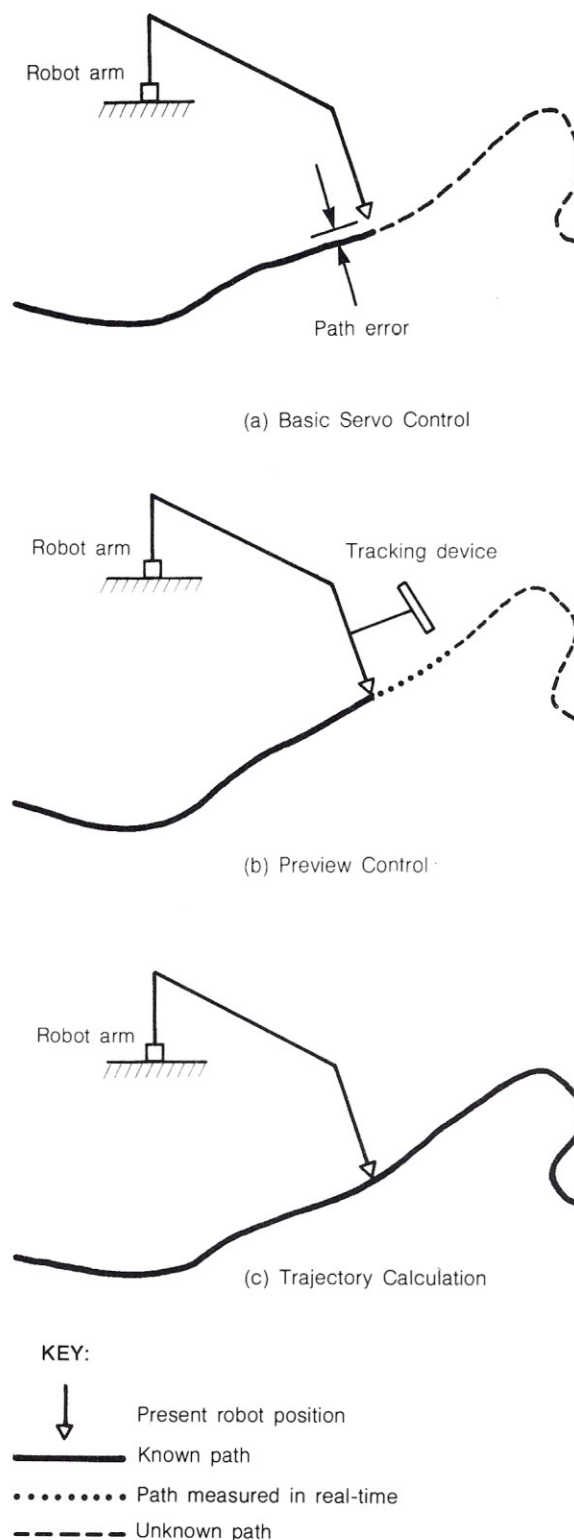


Figure 7. Three categories of continuous path control. In (a), the controller uses present and past error to determine motor drive signals. In (b), future path information is added to improve tracking. In (c), nominal motor drive signals are precomputed for the entire path, and modified in real time if tracking errors occur.

determined, let's say, by the computer's real-time clock, we can measure time by counting loop cycles. In the following discussion, we will assume that all time-varying quantities are measured at the start of some particular interval, designated by an integer (j through m).

The key ingredients used in our calculations for the control of a single robot joint are the position where the joint *should* be at a given time, $x_d(k)$, and where the joint *actually* is or was at some particular time, $x(k)$. For control of a stepping motor, our goal is to compute the appropriate step rate command, V_c , that will correct any tracking error and keep the robot on the desired path.

Readers familiar with stepping motors might wonder why, since the computer is in control of the step rate sent to the motor, there should be any tracking error at all. Why not just compute the number of steps necessary to go to the desired position and send that many pulses to the motor during the next interval?

Unfortunately, it isn't that simple. We may have to allow for missed steps by providing feedback for closed loop control. Alternatively, we must place restrictions on the change to the step rate to stay within the torque limitations of the motor. In this case the *actual* rate we send to the motor may be higher or lower than the desired rate, with the result that the error will not be corrected in one cycle.

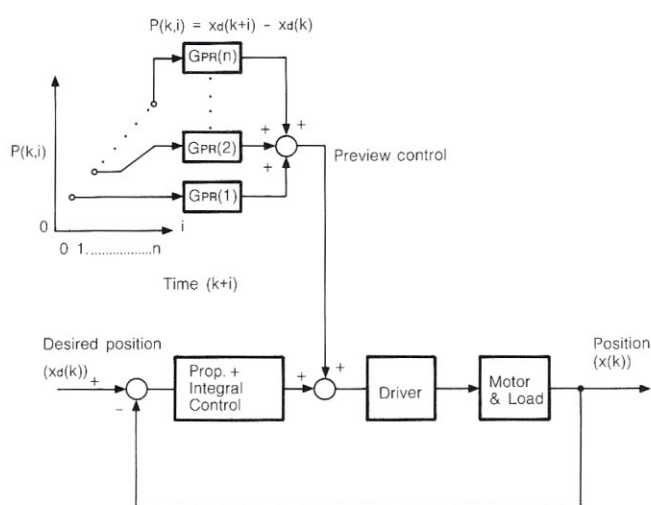


Figure 8. One approach to introducing preview correction terms is to give each preview "error" $P(k,i)$ its own weight, $G_{pr}(i)$.

In [4], Snyder described the basic ideas of "proportional" and "integral" correction terms for correcting tracking errors. For our case of discrete-time control of a stepping motor, a standard "PI" control computation would have the form:

$$V_c(k) = G_P E(k) + G_I \sum_{j=0}^k E(j)$$

where $E(i)$ is the tracking error at the time of the " i "th cycle: $x_d(i) - x(i)$. The proportional and integral gain coefficients, G_P and G_I , respectively, determine the response of the system to errors. They must be "tuned" for the best performance, and their values are influenced by the choice of the loop cycle time, which should be kept constant. The integral term incorporates the influence of past tracking error by summing up all the error amounts since the beginning of the path.

If we have some description of the path ahead, perhaps from vision or tactile sensing, we can compute values x_d for a few intervals beyond the current one and use these for preview calculations. The idea of preview path control is not new—it has been applied to servo motor control for some time. [6,7] We can introduce preview correction into our simple PI stepping motor controller by including preview "error" terms,

$$P(k,i) = x_d(k+i) - x_d(k),$$

into the step rate calculation.

One way of including these terms is to add in several of them, each weighted by its own preview gain coefficient, $G_{pr}(i)$. This tactic is illustrated in Figure 8. The equation for this controller would be:

$$V_c(k) = G_P E(k) + G_I \sum_{j=0}^k E(j) + \sum_{i=1}^n G_{pr}(i) P(k,i)$$

where n is the number of preview samples available. To see the effect of these preview terms, suppose the path for this joint moves in the $+x$ direction on the next few cycles. In this case, the preview terms $P(k,i)$ would be positive. With positive preview gain coefficients, this path change would result in an increased step rate command *before* the point where the path changes. In effect, the controller "anticipates" the change and starts to speed up before a tracking error occurs. Of course, the reverse occurs if the path changes in the $-x$ direction. Instead of a simple PI controller, we now have a "Proportional + Integrating + Preview" (PIP) controller.

Applying the PIP Control Scheme

The hardware we used to test the PIP controller design was a six axis welding robot in our lab at Berkeley. (This machine will be described in more detail in Part II of this article.) Two axes of the robot are realized by an x-y table that moves the workpiece in a plane under the welding tip. The table is positioned by lead screws that are turned by stepping motors. To simplify matters, we limited our experiments to controlling these two motors only. To avoid having to worry about missed steps, we used optical encoders for position feedback on each lead screw. Figure 9 shows an architectural diagram of the experimental setup.

At the time of our tests, no vision or tactile sensing was available to use as the source of preview knowledge for the controller. Instead, we used a representation of the welding path stored in memory, much in the way traditional "blind" industrial robots are programmed. We

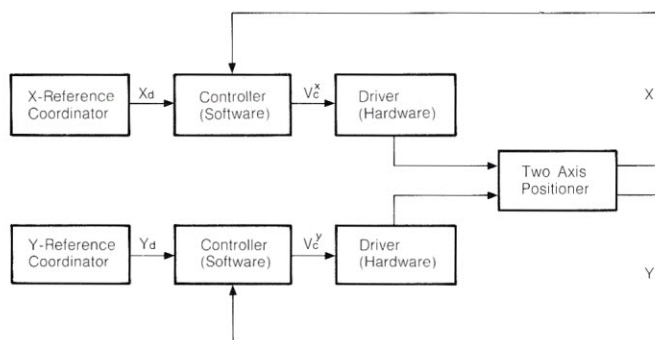


Figure 9. Architecture of the experimental setup used to test the PIP tracking controller.

could simulate the effects of using sensory evidence by superimposing a randomly generated error on the preview data derived from the stored path. This let us test the PIP path control system's response to noisy position measurements typical of those made by real tracking devices.

The problem of applying a feedback control scheme is largely one of choosing the gain coefficients that give the best performance, which, in this case, means getting the robot to accurately track the commanded path. Of course, minimizing the computations necessary to accomplish this in real time can also be a problem. We tried several different PIP control schemes, corresponding to different

Terrapin Turtle

Be one of the first persons to own your own robot. It's fun, and unlike other pets, the Turtle obeys your commands. It moves, draws, blinks, beeps, has a sense of touch, and doesn't need to be housebroken. You and your Turtle can draw pictures, navigate mazes, push objects, map rooms, and much, much more. The Turtle's activities are limited only by your imagination, providing a challenge for users of all ages. Interfaces, including software for easy control of the Turtle, are available for the Apple, Atari, and S-100 bus computers.

Terrapin will give a free Turtle to the person or persons who develop the best program for the Turtle by March 31, 1982. In addition, Terrapin will pay royalties. For more information, write or call;

Terrapin, Inc.
678 Massachusetts Avenue
Cambridge, MA 02139
(617) 492-8816

Books available from Terrapin

Turtle Geometry by Abelson and diSessa

An innovative book using Turtle Graphics to explore geometry, motion, symmetry and topology. MIT Press \$20.00

Mindstorms by Seymour Papert

An exciting book about children, computers, and learning. Explains the philosophy of the new LOGO language. Basic Books \$12.95

Artificial Intelligence by Patrick Winston

Explores several issues including analysis of vision and language. An introduction to the LISP language is incorporated in the second section. Addison-Wesley \$18.95

Katie and the Computer by Fred D'Ignazio

A children's picture book adventure about a young girl's imaginary trip inside a computer. Creative Computing \$6.95

Small Computers by Fred D'Ignazio

A book about the future of small computers and robots, aimed at adolescents. Franklin Watts \$9.95



The Perfect Pet™

choices of the preview gain terms and their relationship to the other gains. For more details on the procedure, see [8].

The control equation that gave the best results, minimizing computation time without sacrificing good path control, was a "modified simple PIP" control scheme that used only one preview data point and ignored integral error effects older than a few cycles. The formula for this controller is:

$$V_c(k) = G_P E(k) + G_I \sum_{j=k-n_I}^k E(j) + G_{pr} P(k, n)$$

where n_I is the number of past samples used for integral correction and n is the number of steps ahead of the current position that the preview sample is taken from. Figure 10 shows the path information used in this scheme.

The first term is the standard proportional correction, the second is a modified integral action term. Integral control action has the effect of minimizing steady state error at the expense of increasing the transient settling time. When the integral term was computed from the start of the path (i.e., from $j=0$), the settling time was excessive. Using only the last n_I terms gave a good compromise between settling time and steady state error when n_I was about 10. The last term in the control calculation is the simple preview correction term. Unlike the earlier expression that summed the effects of several points, this rule uses the n th preview term and ignores the others in between.

The remaining factors to be determined are the three control gain coefficients and the number of steps ahead, n , that the preview term will look. Appropriate gains for servo systems can be derived analytically or by trial and error experimentation. Often a combination of the two methods is used. To simplify matters, we assumed that $G_P = G_{pr}$, and selected a trial value for these terms and the integral gain G_I . I'll talk more about the "tuning" procedure for adjusting these gains in Part II. For now, let's assume that we have already done this, and look at the performance of the controller for different values of n .

Performance of the Simple PIP Control Scheme

We made several test runs using the simple PIP control rule to drive the steppers on the x-y table. The same pre-programmed path was used in each case, consisting of a 10" straight line followed by a 90° turn onto a full circle of 5" radius, so that the straight line formed the diameter of the circle. The robot was supposed to follow this path at a

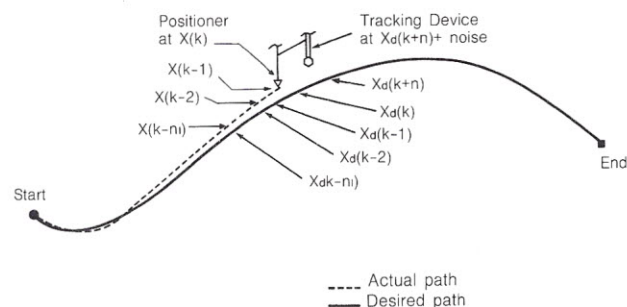
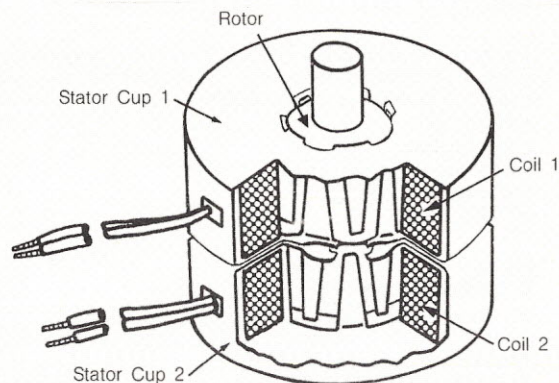


Figure 10. Path information used by the modified simple PIP control scheme.

STEPPING MOTOR DESIGN

Because of their construction, stepping motors feature easy interfacing and simplified control with digital circuitry. The key to stepping motor performance lies in the design of a rotor and stator combination that has regularly spaced equilibrium positions created by alternating magnetic poles. In its most common form, the rotor is constructed of a ceramic permanent magnet that has a fixed pattern of alternating north and south poles and a stator made from two sets of toothed soft-iron cups energized by separate windings. The teeth on the cups are folded so that when energized they form alternating poles that interact with the rotor fields, producing either movement or a holding torque.

The pole patterns created by the two windings are 90° out of phase. Therefore, by selectively switching the windings the motor will either step forward or in reverse. A simple logic circuit is all that's needed to convert pulses from a computer output port to appropriate stepper winding control signals. That circuit, as well as the power circuits to drive the windings of a typical stepper, are described in "A Homebuilt Computer Controlled Lathe," in the May/June issue of *Robotics Age*.



constant velocity. Obviously, this would require an instantaneous velocity change at the 90° turn, so some tracking error was expected. Figure 11 shows three test runs, with the solid line indicating the desired path and the dotted line the path actually followed. In each case, the desired velocity was 2 in/sec. The operating frequency of the control loop was 10 Hz.

In case 1, we used a single step look-ahead ($n=1$ in the equation). Here the performance was similar to a conventional feedback controller, with the sharp turn resulting in a classic "step-response" damped oscillation. In case 2, the preview sample was taken three steps ahead ($n=3$). The actual path showed marked improvement, with only a little overshoot on the turn. When the preview was taken five steps ahead, the controller "saw" the turn five steps ahead, and actually rounded off the corner a little.

In welding, the velocity of the welding tip must be carefully controlled to get the best results. Figure 12 shows plots of the position and velocity errors over the course of the path for the three tests of Figure 11. Position error is defined as the distance between the desired x-y position and the actual one. Velocity error is the magnitude of the

vector difference between actual and desired table velocities, with the sign determined by comparing the actual (scalar) speed with the commanded speed. The first two seconds for all three cases were identical. This is due to the limited acceleration of the steppers to get the table up to the desired speed. The superiority of the PIP controller can be seen when the corner is encountered at 5 sec. into the path. The errors were excessive for Case 1 which, used little preview, but were drastically reduced in Case 3.

We also investigated the effects that changing the desired tracking speed had on the performance of the controller and of introducing noise into the preview measurements. Figure 13 shows the results of a number of test runs made with different values on n and at speeds of 1 and 2 ips. From these we saw that at low velocities, a little preview can reduce velocity error at the expense of increased position error. At higher velocities, preview improved performance significantly unless the controller looked too far ahead. A look-ahead of four steps seemed to be best for our system. Introducing a random noise of .1" average error into the preview measurements increased the tracking error by much less than .1".

KEY:

— Desired path
 Actual path where
 different from desired

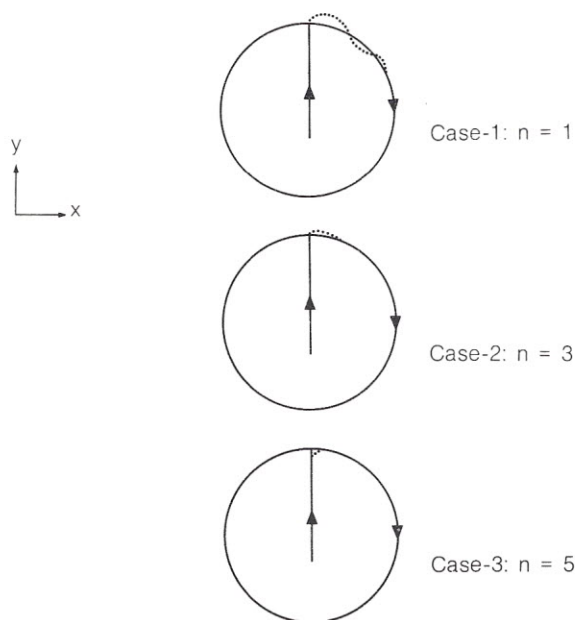


Figure 11. The results of three test runs using different values of n , the number of steps ahead that the preview sample is made.

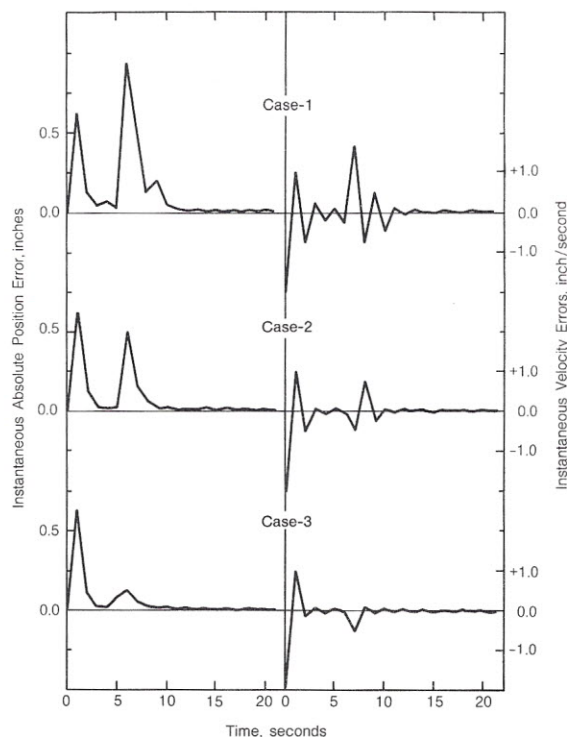


Figure 12. Position and velocity errors for the three test runs of Figure 11.

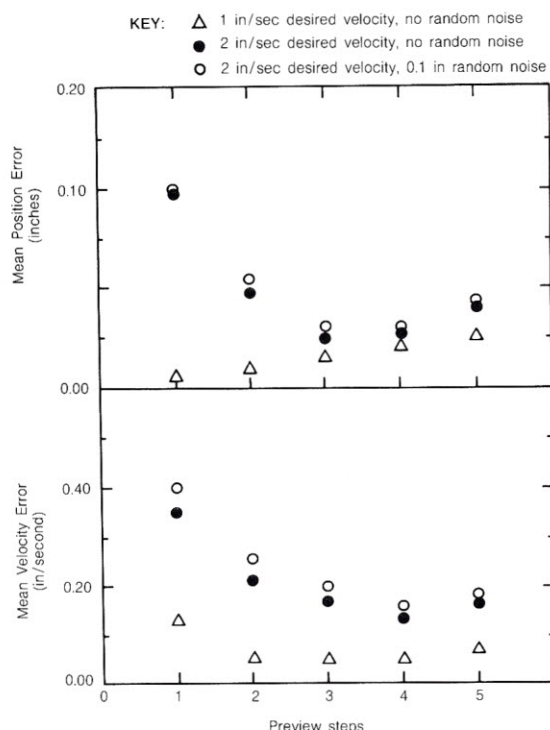


Figure 13. The effects of tracking speed and preview measurement noise on controller performance.

Conclusions

Because they are so easy to interface to computers, stepping motors are getting a good deal of attention for use as actuators for robots. When some sort of position/velocity feedback is available, a stepper can be used like a servo motor—without the analog power amplification that servo motors usually require.

The real advantage possible with steppers is the elimination of the requirement for getting direct feedback from the controlled mechanism. If the mechanical loading and inertia are well defined, proper attention to the limitations of the stepping motor can keep it from ever missing a step (provided that collisions are avoided), and position can be measured just by counting output steps. (The motor's physical characteristics need not be quantitatively known, just well-behaved, so that satisfactory constants can be obtained experimentally.)

Regardless of the architecture used, the PIP control scheme described here gives a stepper-driven robot the ability to track a continuous path accurately. A key advantage of the preview scheme is that it can make use of measurements not only from a pre-programmed path, but from path sensors such as vision, touch, or other seam-tracking devices.

In this article I hardly mentioned the actual hardware and software design of our robot and its control system. I will give a more complete description of this in Part II. Work on improving the performance of the PIP controller and the stepping motor control circuitry is still in progress at Berkeley's Mechanical Engineering Department.

References

- [1] Porter, J., "Stepping Motors Move In," *Production Engineering*, vol. 34, February, 1963, pp 74-78.
- [2] Kiebertz, B. R., "The Step Motor—The Newest Advance in Control Systems," *IEEE Trans. on Automatic Control*, January, 1964.
- [3] Kuo, B. C., *Theory and Application of Step Motors*, West Publishing Co., 1974 Ed.
- [4] Snyder, W., "Microcomputer Based Path Control," *Robotics Age*, Spring 1980.
- [5] See discussion in the article "Robotics Research in Japan," by K. Takase, *Robotics Age*, Winter 1979-80.
- [6] Tomizuka, M., Whitney, D. E., "Optimal Discrete Finite Preview Problems (Why and How is Future Information Important?)," *Trans. of ASME, Journal of Dynamic Systems, Measurement and Control*, Vol. 97, No. 4, December, 1975.
- [7] Tomizuka, M., Dornfeld, D., Purcell, M., "Application of Microcomputers to Automatic Weld Quality Control," *Trans. of ASME, Journal of Dynamic Systems, Measurement and Control*, Vol. 102, No. 2, 1980.
- [8] Dornfeld, D., Tomizuka, M., Motiwalla, S., Tseng, R., "Preview Control for Welding Torch Tracking," *Proc. 1981 Joint Automatic Control Conference*, University of Virginia, June 1981.
- [9] Engelberger, E. F., *Robotics in Practice*, Amacom publication, 1980 edition.

ANNOUNCING...

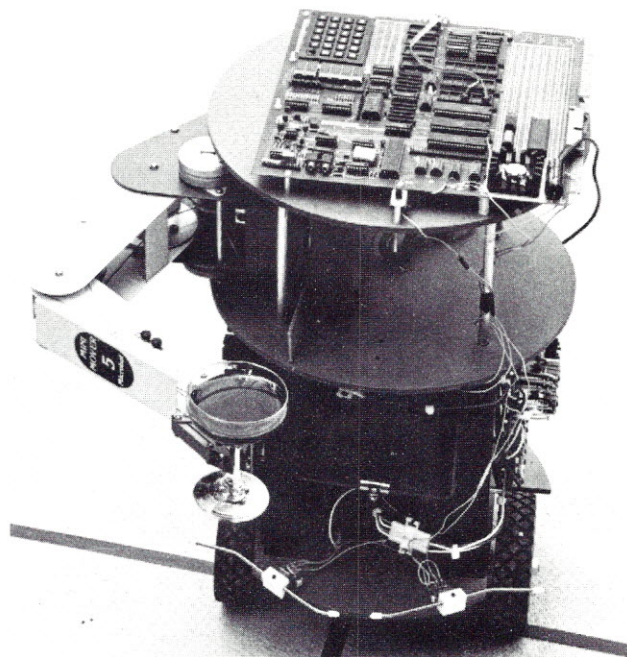
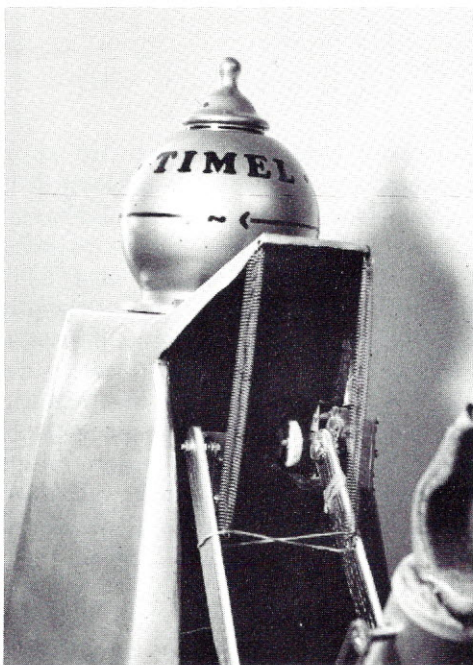
Robotics Age Home-Robot Photo Contest

Proud of your home-built robot? Send us three photographs of your robot, along with a description, and you may win your choice of a life-time subscription or \$150 and a two-year subscription to **Robotics Age** Magazine.

FIRST PRIZE—Your choice of a life-time subscription (worth \$250) or \$150 cash and a two-year subscription to **Robotics Age** Magazine.

SECOND PRIZE—\$50 and a one-year subscription to **Robotics Age** Magazine.

THE WINNING PHOTOGRAPHS will be gloriously published in the magazine!

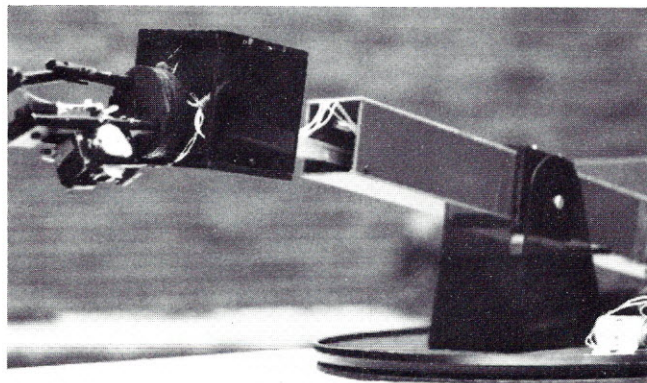


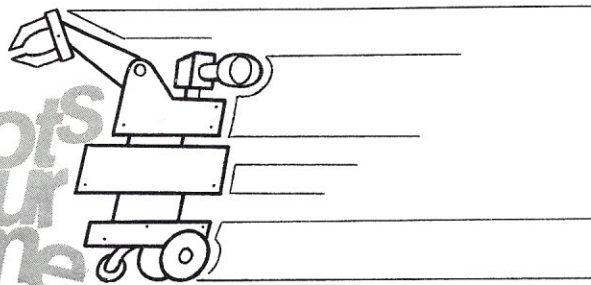
Here's how to enter:

- Send us at least three quality photographs and a description of your home-built robot. One photo should show an overall view of the robot's exterior, one should show the robot's mechanical details and/or circuitry, and one should show the robot performing some task.
- Remember to include several pages describing the construction and capabilities of your robot (including software, which we can't see in the photo! Be sure to emphasize any design innovations you've developed, within the photos and in the description.
- All entries should be sent to **Robotics Age**, P. O. Box 801, La Canada, California 91011.

And don't forget these Rules:

- The contest ends October 1, 1981.
- Only computer-controlled robots qualify.
- All photographs submitted become the property of **Robotics Age** Magazine.
- If you submit an article on your home robot to us, accompanying photographs are automatically eligible to win the contest!
- Winners will be chosen on the basis of the novelty of their robot, the complexity of the task it performs in one of the photographs, and the quality of the photographs themselves.





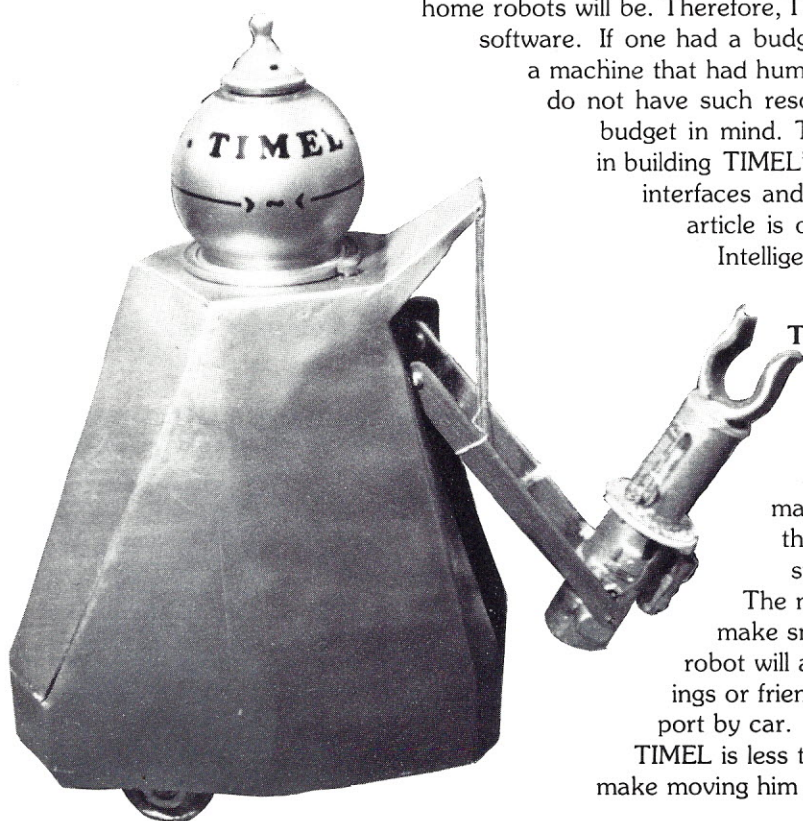
by
John Blankenship
Devry Institute of Technology
Atlanta, Georgia

TIMEL: A Homebuilt Robot

PART 1

The building of my robot, TIMEL, began with an assumption: hobbyists who want to contribute to the emergence of intelligent home robots should concentrate their efforts on software. It might be expensive, but we can build hardware theoretically capable of opening doors or washing windows.

We can give robots the parts needed for voices, ears, and even eyes. But it is software—intelligent control programs—that will determine what these pieces can do and how useful home robots will be. Therefore, I built TIMEL for the purpose of developing intelligent software. If one had a budget like NASA or DoD, it might be best to start with a machine that had human strength and dexterity. Unfortunately, most of us do not have such resources. Most hobbyist systems are built with a low budget in mind. Therefore I tried to use inexpensive everyday items in building TIMEL's body. I wanted to save my major investment for interfaces and computing power. The hardware discussed in this article is only a vehicle for developing the software for Truly Intelligent Mechanical Electrical Life—TIMEL.



The Base and Body

A robot designed as a medium for software development does not have to be the six foot man-like structure we have come to expect. In fact, there are many reasons for going in the opposite direction. Since the system will be testing experimental software, its strength and size should prevent human intervention.

The machine should not be able to do more damage than make small scratches on the furniture and walls. A hobbyist robot will also be expected to make appearances at club meetings or friends' houses and, therefore, should be easy to transport by car.

TIMEL is less than three feet tall, including a head that detaches to make moving him easier. A seat belt holds the robot snugly in the car.

As shown in Figure 1, TIMEL's base is made of plywood and measures 18 inches wide by 22 inches from front to back. The motorized wheels, shown in Figure 1, lift the base 7 inches off the floor. To keep a low center of gravity, the battery sits on an aluminum bracket that hangs from the base.

Both wheels can be independently driven in forward or reverse—and therefore provide both power and steering. With one wheel in reverse and the other moving forward, TIMEL gracefully pirouettes. (I purchased the wheels from Herbach & Rademan, 401 East Erie Ave., Philadelphia, PA 19134.)

To give the front caster a sturdy mount, the cradle for TIMEL's arm is also constructed of wood. Thin wooden skirts wrap around the base, making it seem lower than it is. These skirts will later serve as the place to mount fail-safe bumper switches.

To keep TIMEL's weight down, I built the entire body-shell out of cardboard. Its construction was fairly simple: First, I cut cardboard panels into the shapes shown in Figure 2, then taped and glued them to form TIMEL's front shell and removable rear panel. Next, I dipped six-inch squares of fiberglass cloth in resin and used them to laminate the entire body. The front shell was fiberglassed first. To make the removable rear panel fit exactly, I covered its edges with sandwich wrap (this prevents sticking), pressed it into place against the front shell, and fiberglassed it while in place.

A little automobile putty, a lot of sanding, and two coats of paint later, TIMEL started to take on some personality. I heartily recommend using cardboard and fiberglass if you are even a little handy. I had never worked with fiberglass, yet my first effort produced reasonable results.

TIMEL's head is an example of using common objects for robot construction. It is made of an inverted fishbowl, topped with a candy-dish lid. The surface is spraypainted with a light coat of metallic paint. Though normally opaque, the bowl glows nicely when lit from inside.

You can be sure that when you build a robot for software development, you will, at some time, need to modify or repair it. The robot should therefore be designed so you can easily access its internal workings. There are two ways to do this: make the robot shell removable or provide a way to pull the circuitry outside the robot. I chose a combination of the two. TIMEL's rear panel slips off, leaving very little covered. And all major circuitry is mounted on the battery cover, which you can pull out without disconnecting the ribbon cable. Figure 4 exposes TIMEL's insides.

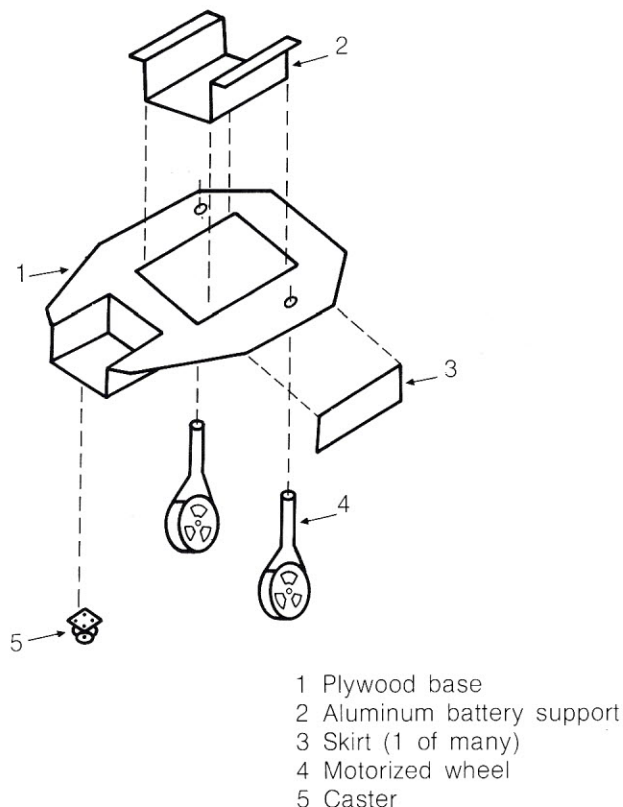


Figure 1. TIMEL's base.

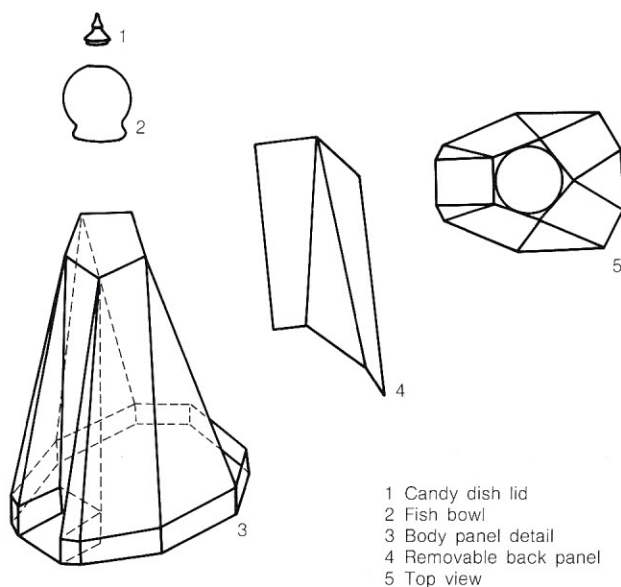


Figure 2. The body-shell.

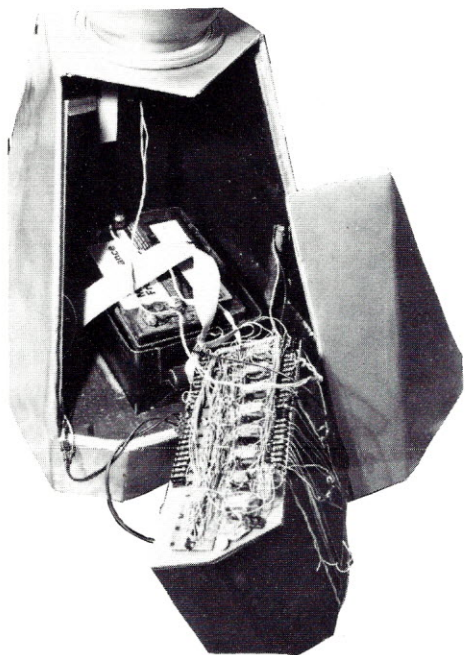


Figure 4. TIMEL's inner workings.

The Manipulator

In building TIMEL's manipulator, I decided to trade off a certain amount of strength and agility for low cost and simplicity. The manipulator has four degrees of freedom, rather than the traditional six: TIMEL's base rotates, its shoulder elevates the arm, its elbow flexes, and its wrist rotates. This simplicity allowed me to build TIMEL's arm out of humble materials which, though they fill their purpose well, may be overlooked by many hobbyists.

TIMEL's forearm, for example, needed to be lightweight and yet support three motors. The elbow consists of a section of tube from the center of a roll of carpet. Fitting snugly into the tube, a potato-chip can forms the wrist. The cardboard-to-cardboard bearing—with a little vaseline added for smoothness—may not pass an industrial inspection, but with lightweight objects, it works efficiently and never seems to wear down. Figure 5 shows TIMEL's forearm.



Figure 5. The rotating forearm/wrist assembly.

In Figure 6, you can see the arm in more detail. All the motors shown were found in junk stores. Almost any motor will do; and most can be fitted on the arm by simply cutting out the proper sized holes and epoxying the motors in place. I recommend gearhead motors—those with an appropriate gear train attached. They are usually lighter and easier to mount.

Referring to Figure 6, motor 1 rotates the elbow, motor 2 controls the shoulder (two coil springs help carry the weight of the arm) motor 3 rotates the hand, and motor 4 opens the fingers. The large gear that rotates the wrist—labeled 5 in the figure—was made from a wooden "donut" to which I glued a plastic strip of gear-like teeth (around the circumference of the annulus). Again, scavenging for parts, I found the strip of teeth with a toy airplane. (When the teeth are inserted and quickly withdrawn from the front of the plane, it spins the propeller.)

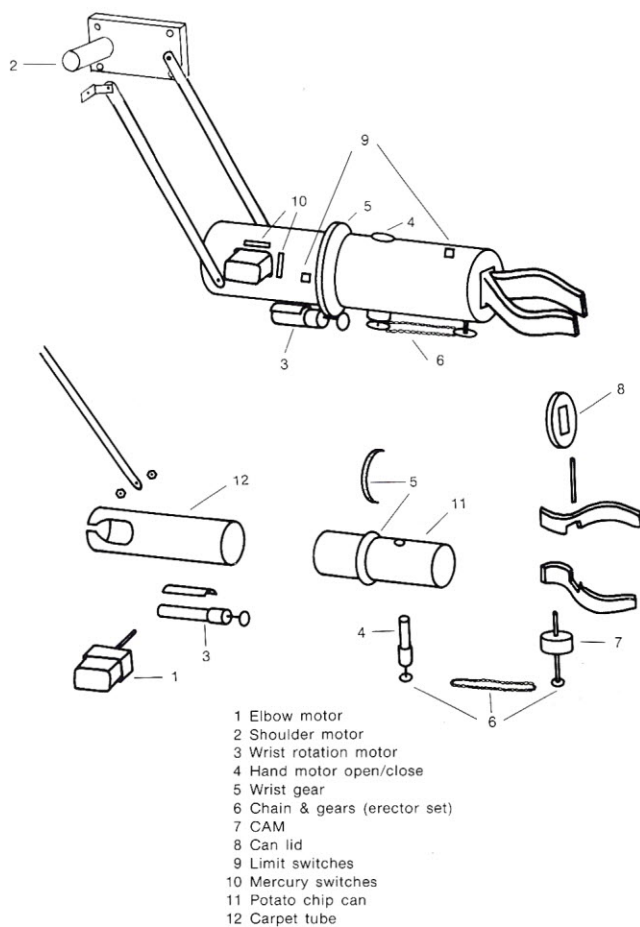


Figure 6. The construction of TIMEL's arm.

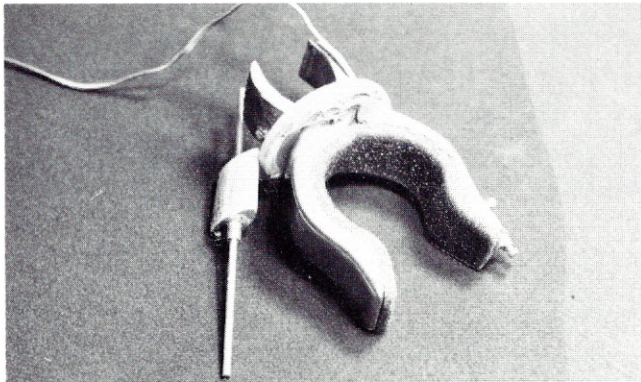


Figure 7. TIMEL's fingers.

After experimenting with many styles of grippers, I made TIMEL's hand from a short length of 2x4 pine board. I used a jig saw to cut out two S-shaped pieces and then fit them together much like a pair of pliers. As Figure 7 shows, a strip of foam rubber is glued inside the fingers to improve TIMEL's grip. A spring, not shown in that figure, holds the jaws in their normally closed position. Lying beside the gripper in Figure 7 is a wooden cam. Through a chain and sprocket drive, motor 4 turns the cam which in turn opens the robot's fingers. You may have noticed that two small switches are mounted on one of the fingers. They connect to TIMEL's onboard logic and allow for automatic override—which I'll say more about in Part II of this article.

Conclusion

A home robot does not have to bankrupt the hobbyist. TIMEL proves that by using common and inexpensive objects, you can build an adequate testbed for robot software development. This lets you put your money into computational power, rather than mechanics.

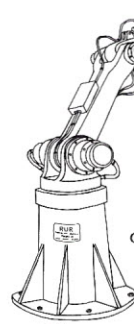
In Part II of this article, we'll take a look at TIMEL's control electronics.



Figure 8. TIMEL pours his master a stiff one.

R OBOTS

The New Industrial Revolution



Intelligent machines are rapidly appearing in factories, offices, homes, and automobiles. The face of the industrialized world is changing. Vast opportunities exist for those with a knowledge of this explosive field. **Robotics Age** will bring you the latest developments in **all aspects** of Robotics and Machine Intelligence. From introductory and non-technical survey articles to the latest state-of-the-art in hardware and software design. You will find articles on sensors and perception, computer reasoning and robot control, industrial application and the automated factory of tomorrow—as well as related new products, book reviews and technical abstracts.

YES! I want to stay up-to-date on this fascinating new technology!

Name _____ Title _____

Company _____ Address _____

City _____ State/Province/Country _____ Postal Code _____

	United States	Canada Mexico*	Foreign Rates*
☐ 1 year (6 issues)	\$15	\$17	\$19
☐ 2 years (12 issues)	\$28	\$32	\$36
☐ 3 years (18 issues)	\$39	\$45	\$51

*US Funds on US Bank
☐ Bill VISA ☐ MasterCard ☐ Bill me (N America only)

Card Number _____ Expires _____

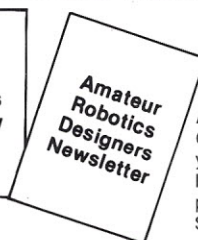
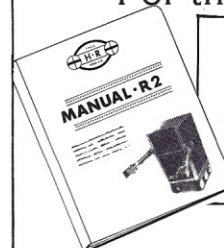
Signature _____

Send to: **ROBOTICS AGE**, PO Box 512, Tujunga, CA 91042

CIRCLE 22

HOBBY ROBOTICS' Shoppers Mart

— For the SERIOUS Hobbyist —



ARD Newsletter. Quarterly. Keeps you abreast of latest news, perks up interest! \$6.00

Complete RU-2 Manual! Everything you need to build your own robot shown, detailed! Build from base up with Hobby Robotics' highly maneuverable 2-ft. per sec. super-traction base, extendable arms lift-rated at 15 lbs., grip-like hands. Manual only \$25.00, refundable with first purchase.

Hobby Robotics' Parts Supplier Directory. Chock-full of names, addresses, phone numbers you want for hard-to-find parts, systems, build-your-own-robot info. Only \$3.50. Get yours now!

HOBBY ROBOTICS

5070 Buford Highway
Norcross, Ga. 30071 404/448-0190

Please send items checked below:

☐ Qrt. Newsletter ☐ RU-2 Manual

☐ Parts Directory ☐ Robot Base

FREE ☐ Hobby Robotics' Catalog **FREE**

Total Enclosed \$_____ (includes shipping)

☐ Personal Check ☐ Cashiers Check/M.O.

☐ Mastercharge ☐ Bank Americard

Acct. # _____

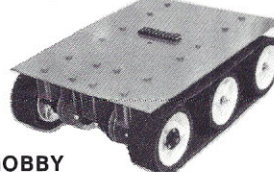
Signature _____ Exp. _____

Name _____

Address _____

City _____

State _____ Zip _____



HOBBY ROBOTICS' ROBOT BASE
Ready to roll!

ONLY \$495 (includes shipping)

Eye to Industry

by J. W. Saveriano

News from Abroad:

AUTOMAN '81, the first European Automated Manufacturing Exhibition and Conference, was held on May 19-21 in Brighton, UK. The conference was jointly sponsored by the **British Robot Association (BRA)**, the **Production Engineering Research Association (PERA)**—England's prime promoter of robotics, the **Institute for Production and Automation (IPA)** of Stuttgart, the **French Industrial Robot Association (AFRI)**, and the **Italian Industrial Robot Association (SIRA)**.

Britain's **Margaret Thatcher** and Unimation's **Joseph Engelberger** were keynote speakers. Strong words—"Automate today or stagnate tomorrow"—were the theme of the conference. Apparently, the Europeans are anxious to compete with the robotic factories of Japan, West Germany, and the United States.

AUTOMAN's exhibition emphasized hardware for assembly, most of which is familiar in the US: simple pick and place devices and rotary table multi-station assembly machines, for instance. Several robots were shown. Among these was the Pragma A3000 Assembly robot built by **Digital Electronic Automation (DEA)** of Italy. A Cartesian robot with a horizontal arm, the Pragma A3000 can selectively pick up pre-ordered parts, perform dimensional control, assemble groups of components according to their tolerances, and test assembled products, grading them according to predetermined quality categories.

Systems Control demonstrated a prototype mini-arm called the Smart-Arms 61. The 6-axis arm has a capacity of 2kg and a maximum reach of 600mm. Its positional resolution was given as 1mm. Consett Engineering will manufacture Smart-Arms 61 in England and expects to sell it for £5,000 (roughly \$10,000).

Also at the AUTOMAN exhibition was a prototype pneumatically operated robot hand. The advanced 5-fingered hand is being developed by **Cranfield Robotics and Automation Group (CRAG)** and the **Science Research Council (SRC)**, both of Britain.

Turning to France, the *Wall Street Journal* expressed some uneasiness about robotics' future in that country. The previous president, **Valerie Giscard d'Estang**, had begun a program to move France out of such industries as steel and shipbuilding and into areas such as robotics. His government was allotting more than \$300 million dollars to high-technology projects. The *Wall Street Journal* predicts that "A **Mitterand** government probably wouldn't pursue

this industrial policy. The Socialists are committed to creating more jobs and to reviving steel, textiles and other traditional industries."

At present we can only speculate about France's new attitude toward robots. But without a doubt, the **People's Republic of China** has taken an ideological stand against robots. In the *Beijing Review*, an English language "Chinese weekly of news and views," the question was asked, "Can Robots Create Value?" Here in part is their answer:

"According to the Marxist labour theory of value and the theory of surplus value, robots and automatic hands are constant capital and cannot create value. In so far as they have value themselves and gradually transfer the value to products...they constitute an element of the value of their products but do not add any value to them. The only source of new value and surplus value is living and hired labour."

According to this view, robots merely transfer the value that human labor puts into their construction. (This ignores the potential for self-replicating, self-repairing robots.) The *Beijing Review* contends that "no matter how technically advanced the robots and automatic hands are, they are still manufactured, operated, controlled, maintained and improved by man. Without man, they remain dead junk."

Unlike the Chinese, West Germany is more concerned with applying robots than ideology—especially since the Japanese have been capturing such a large share of their domestic automobile market. With this in mind, the **West German Ministry for Research and Technology** is now supporting robotic research in several laboratories. Among the major areas of study are welding and assembly applications. In work similar to that of **SRI**, television cameras are used as sensors to monitor the arc and molten pool while welding, providing realtime data for adaptive control of the manipulator and torch.

For applications in automatic assembly, the West Germans built a five-axis compliant-gripper suspension module. Using elastomer springs, the gripper resembles the remote-center compliance device developed by **Dra-per Laboratories** in Massachusetts.

Finally, from a joint press release from Stockholm, I learned that **ASEA** and **Electrolux**, two of Sweden's largest companies, have begun talks on merging their industrial robot operations. Until now, ASEA has specialized in

sophisticated electric drive robots while Electrolux has concentrated on robots using the simpler pneumatic drive. Both companies build high quality robots and should benefit from the merger through the broadening of their product lines.

Meanwhile back in the States

While many US universities are doing long-term basic robotics research, **Rensselaer Polytechnic Institute** (RPI) hopes to entice some good students to work on manufacturing problems. As George Low, RPI's president laments, "Sixty-eight percent of the wealth production in this country comes from the manufacturing industry. Yet today only a small fraction of our research and development goes into manufacturing engineering. The best people are not attracted to manufacturing related jobs. It is not exciting to them."

Under Low's direction, RPI has founded the **Center for Manufacturing Productivity**. From a list of about one hundred manufacturing problems, the Center will select the ten best and assign them to student researchers. For example, One problem already selected asks, "how can we use computer controlled robots to increase assembly-line productivity?"

Unlike most research programs, RPI's Center does not accept government funding. Turning to industry, the program has already received \$500,000 from **General Electric** and \$250,000 from **General Motors**.

Speaking of GM, the **General Motors Research Laboratories** (GMRL) and the **General Motors Manufacturing Development Department** (GMMD) have introduced a new vision system. Called "Keysight," the system recognizes the pattern of properly assembled cylinder head valve springs. From its overhead position on an engine production line, Keysight can spot faulty assemblies—such as missing retainer keys in the cylinder heads.

GM has contracted **Robogate Systems, Inc.**, to install two "Robogate" automated body-framing systems. (Robogate Systems is the American licensee of Comau Industrial a subsidiary of Fiat Motors). These will be used in producing the J-cars at GM's Janesville, Wisconsin, assembly plant. From 35 to 40 spotwelding robots will be used in conjunction with the Robogates. The suppliers of these robots are still not known, but sources tell us that they may be a combination of Unimation and Cincinnati Milacron models.

Meanwhile, General Electric has been trying to make up for lost time. During the 1960's and '70's, when they were

not aggressively pursuing advances in automation, they fell behind their Japanese competition. Now, GE is voraciously gobbling up companies in acquisitions, expanding facilities, and signing licensing agreements to use, build, and sell other companies' robots. For instance, GE has recently spent \$100 million dollars to complete its acquisition of **Calma Company**. Located in Sunnyvale, California, Calma manufactures interactive graphics systems for use in computer-aided design and manufacturing (CAD/CAM).

GE has also recently signed two important agreements with firms outside the US. One is a licensing agreement with the Italian robot company, **DEA**. GE has obtained the worldwide rights to manufacture and sell DEA's Allegro assembly robot system. Another agreement, with Hitachi, calls for a general technical exchange. GE will receive information on robot technology and mechanical parts assembly systems.

A three-year program jointly funded by NASA and industry has been announced. Funded with about \$500,000, the program has the **Jet Propulsion Laboratory** (JPL), **Unimation**, and the Hamilton Standard division of **United Technologies** among its participants. Its goals are to make industrial robots more productive and safer to operate, and to reduce tooling costs. Using infrared range-sensing technology developed at JPL's Robotic Group, this program is part of NASA's effort to apply its technology in industry.

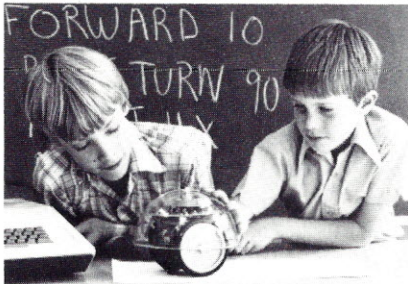
Another example of research that may benefit industry can be found at **Louisiana State University**. There, a team of orthopedic surgeons, physical therapists, and mechanical engineers are putting together a complete mathematical model of the human hand. This study could affect the design of grippers in the future.

Finally, let's take a small peek at the future: A report issued by **International Resource Development** (IRD) in Norwalk, Connecticut, claims that **IBM**, **XEROX**, and **Matsushita** will introduce typewriters with voice recognition by 1983. IRD predicts that the typewriters will correctly identify 93% of the "typical business English as spoken by the average executive."

Be seeing you!

If you have robotics-related information that you believe is noteworthy, or have suggestions for topics for this column, please write us at *Eye to Industry*, **Robotics Age**, PO Box 725, La Canada, California 91011, or call 213/352-7937.

NEW PRODUCTS



Turtle Now Has Apple Interface

Terrapin, Inc. announces that it now has a smart Terrapin-Apple Interface for its robot, the Turtle. The interface enables the user to control the Turtle from a high level language (Basic, Pascal, LOGO, etc.) via I/O statements.

The smart interface includes a parallel port, a separate regulated current-limited power supply, and

interface software. The parallel port I/O card contains the interface software in ROM, and plugs into one of the Apple's Peripheral Interface Connectors. The program in ROM initializes the I/O port, generates the appropriate bit pattern for a specific command, compares this with the previous command byte to determine which bits must be left on, and writes to the appropriate output port location.

The Terrapin Turtle costs \$399.95 in kit form, \$599.95 assembled. The Terrapin-Apple Interface with Turtle Talk™ Software sells for \$199.95.

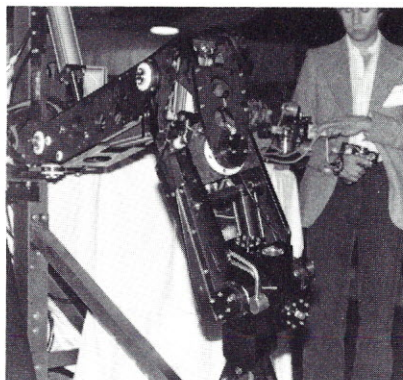
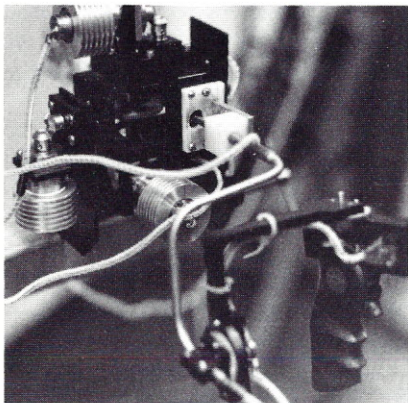
Terrapin Turtles (assembled or kits) and Terrapin-Apple Interfaces (assembled only) are available from: Terrapin, Inc., 678 Massachusetts Ave., Rm. 205, Cambridge, MA 02139, 617/492-8816. **CIRCLE 1**

Hydraulic Manipulator Has Force Feedback

International Submarine Engineering (I.S.E.) Ltd. offers a new manipulator with spatially correspondent or force feedback control. They also now offer "resolved motion" master control arms as well as computer operated arms. Control electronics include software configurable, industry

standard RS232 serial computer ports, allowing the manipulator to be operated as a robot to do repetitive tasks under external computer control. The new manipulator is added to I.S.E.'s line of more than 50 hydraulically powered manipulators, built for underwater vehicles.

Contact: James R. McFarlane, President, I.S.E., 2601 Murray Street, Port Moody, B.C., Canada V3H 1X1, 608/931-2408. **CIRCLE 2**



"X" Axis Transporter

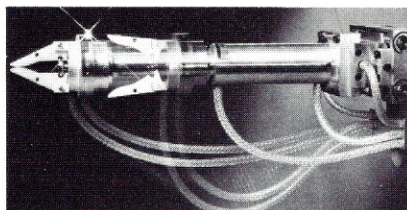
The Mack Corporation adds another product to its family of miniature B·A·S·E robotic components. This product is known as the "X" Axis Transporter, basically a single ended, double acting cylinder with special modifications for installations that must transport small loads over short distances. It can be mounted to an existing transfer mechanism, to standard brackets and to "Y" and "Z" Axis Transporters. Note that the "X" Axis Transporters will operate in any plane. The "X" Axis designation is assigned to this product in order to differentiate between other translating motions in the B·A·S·E Robotics System.

Features include standardized mounting for interchangeability with other B·A·S·E components, a choice of fixed or adjustable travel stops and a choice of free or antirotating (keyed) pistons.

Mack Corporation Series 2 grippers will mount directly to Series 2 transporters, together with other forms of load handling equipment such as vacuum cups, electromagnets, small power tools, measuring equipment and miscellaneous devices requiring transport motion. Operation is positive with pressure to extend and pressure to retract. Maximum operating pressure is up to 300 PSI in either hydraulic or pneumatic service. However, most applications operate on plant air at 80 PSI using a standard four way valve controlled by a sequencer or programmer. At 80 PSI, extending force is 90 lbs. and retracting force is 60 lbs. Force is proportional to pressure and may be adjusted by adding a regulator to the supply line. Payload is 5 lbs., repeatability is within $\pm .010$ and

travel is from 2/10" to 4". Models with anti-rotation key will maintain angular position within 3° when loaded with the rated torque of 10 inch pounds.

Cycle rate is primarily a function of supply pressure, pressure drop and other factors related to control. A practical objective is one cycle per second, which provides approximately seven million transfers per year based on eight hour days and five day weeks.



Series 2A & 2B transporters are available in a choice of travels up to 4" in increments of 2/10". Series 2C and 2D transporters incorporate a manual adjustment which limits piston travel to any station between 0" and 4".

A hybrid installation is available which consists of selecting a Series 2A or 2B transporter with travel longer than required and limiting travel with an internal tube.

For more information, contact Elaine Mack, V.P., Mack Corp., P. O. Box 1756, 3695 East Industrial Dr., Flagstaff, AZ 86002; phone: 602/526-1120. **CIRCLE 3**

Catalog Features Cylinders

Mack Corporation offers a 37 page catalog that serves as a handbook to users of Lo-Profile air and hydraulic cylinders. Over 7000 products are defined in sufficient detail to provide application decisions at a glance. Lo-Profile cylinders range in size from one half

to six inches O.D. and with travels to one inch in increments of one tenth inch. Fluid service is either air or oil with pressure ratings from 150 to 1500 PSI and with temperature ratings to 400° F. Many product lines have been upgraded to include a choice of units in the metric system. Contact: Elaine Mack, V.P., Mack Corp., 3695 East Industrial Dr., P. O. Box 1756, Flagstaff, AZ 86002, 602/526-1120. **CIRCLE 4**

Image Digitizer System

Hinds International announces the Periphicon 500 Series Image Digitizer System. The 500 Series includes the 521 Image Digitizing Camera (32 x 32 pixels) and the 502X Image Digitizer Controller. The controller lets the user interface Periphicon 521 Digitizing Cameras to Digital Equipment Corporation's LSI-11 and PDP-11 architectures (using either Q-BUS or UNIBUS) through the DRV-11 or DRV-11C.

The Periphicon 500 Series System is available in three configurations and can control up to six 521 Digitizing Cameras.

Each control channel of the 502X Controller is independent of any other channel. For each digitizing camera, grey scale (up to 15 levels available) and exposure time are fully under software control. Minimum image acquisition time is 2 msec. Maximum pixel interrogate rate is

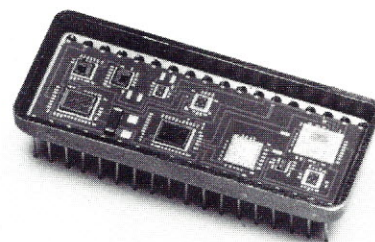
110,000 pixels per second for the Q-BUS.

Since power is drawn from the host, the 500 Series does not require a separate power supply.

Price: The minimum Periphicon 500 System Configuration consisting of a digitizer and an interface starts at \$2,500.

For more information contact: Jimmie Moglia, Hinds International, Inc., P.O. Box 4327, Portland OR 97208, 503/234-7411; Telex: 36-0259. **CIRCLE 5**

Digital-To-Synchro/Resolver Has 1 Arc-Minute Accuracy

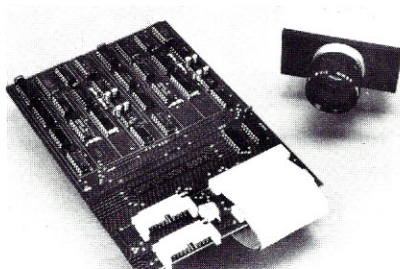


Natel Engineering Co. offers the HDSR2504, a 4-quadrant multiplying digital-to-sin/cos converter.

The converter has a 1 arc-minute accuracy, $\pm 0.05\%$ maximum radius accuracy (scale factor variation), and a resolution of 14 bits. By pin programming the converter can also provide the angular information in synchro format.

Packaged in standard 36-pin DDIP, the HDSR2504 requires ± 15 V power supplies for its operation. The digital inputs are TTL and CMOS compatible. Reference voltages of either 15, 26 or 1.3 V-rms are selected by pin programming, without any requirement for external resistors. For non-standard reference voltages, the converter can be programmed using two external resistors.

Model HDSR2504 converters are



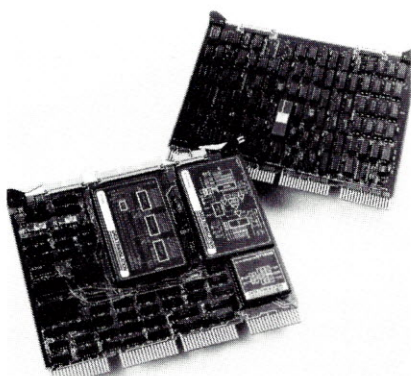
available with angular accuracies of 1, 2 and 4 arc-minutes. These accuracies are guaranteed over the specified operating temperature range. Converters can be ordered with operating temperature ranges of 0° to 70°C, -25° to 85°C and -55° to 125°C.

Pricing in 1-9 quantities is \$445 and availability is stock to 6 weeks.

For further information contact: Natel Engineering Co., Inc., 8954 Mason Ave., Canoga Park, CA 91306, phone: 213/882-9620.

CIRCLE 6

.25 MHZ Q-BUS A/D Card



A new pair of quad-height, microcomputer interface cards featuring data acquisition-to-memory transfer rates of up to 250kHz is now available from Data Translation. The A/D card interfaces through the DEC LSI-11 Series Q-BUS and an on-card multiplexer (MUX) channel file.

The DT3362 includes a high-level, 12-bit data acquisition system (DAS) on a standard quad-size Q-BUS interface board.

Supported by the new DT3369 Dual-Ported RAM Card, the DT3362 can transfer A/D data to memory via an external bus without halting the processor or tying up Q-BUS.

The standard conversion rate is

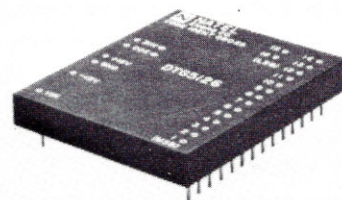
16-Bit Digital Vector Generator

Natel Engineering Co. also announces the DTG5126, a Digital Trigonometric (Vector) Generator. This 16-bit, 4-quadrant, multiplying digital-to-sin/cos converter accepts a digital input of up to 16 bits (CMOS or TTL compatible) and multiplies it with an analog input voltage to generate the products of the analog voltage and both the sine and cosine of the digital input angle. Typical applications for the DTG5126

include coordinate conversion and PPI (sweep) display generation.

For further information contact: Natel Engineering Co., Inc., 8954 Mason Ave., Canoga Park, CA 91306, phone: 213/882-9620.

CIRCLE 7



50kHz, with 250kHz optional. The standard 16SE or 8DI channels can be optionally expanded on-board to 64SE or 32DI, with the configuration specified at time of order.

There is on-card 1024 byte random access memory (RAM) channel list file organized as four independent pages. This feature allows software selection of up to 64 different channels in any desired order. Any sampling sequence up to 256 entries long can be accommodated. Up to four sequences may be stored in RAM and selected under software control.

A two-bit gain select word accompanies each channel select word to give software-controlled gains of 1, 2, 4 and 8 systems having the PGH programmable gain option. This enables the program to continuously set individual channel gains with A/D throughput up to 200kHz.

The DT3362 features seven software-selectable conversion modes which accommodate virtually all standard program- and interrupt-driven input/output (I/O) structures and techniques.

Capable of both stand-alone and DT3369-supported operation, the DT3362 accepts high level inputs of

$\pm 10V$, $\pm 5V$, 0 to 10V and 0 to 5V.

Operating from a single, non-critical 5Vdc supply, the DT3362 includes all provisions needed to program and configure the board as a Q-BUS peripheral.

The DT3369 Dual-Ported RAM Card with controller functions as a 16K-word (32K-byte) RAM containing two independent read/write ports. One port interfaces with Q-BUS, the other port connects directly to the external port of the DT3362 data acquisition system.

Both models are furnished with comprehensive user manuals and diagnostic software routines.

Prices for the DT3362 in 1-9 quantities range from \$2,395 (16SAE/8DI) to \$2,595 (64SE/32DI). The high speed "H" option (250kHz) adds \$1,100 and the "PGH" option (programmable gain) adds a further \$175.

In the same quantities, DT3369 pricing ranges from \$1,675 (15kW capacity) to \$2,775 (64kW capacity).

Availability for all DT3362 versions is 5 days ARO. The DT3369 is scheduled for June shipment.

Contact: Andrew Davis, Director of Mktg., 100 Locke Dr., Marlboro, MA 01752, phone: 617/481-3700.

CIRCLE 8

Smart Ware App-L-LISP

Datasoft, Inc. introduces App-L-LISP Ver 1.7—a major subset of the INTERLISP environment. System requirements are Apple II or II Plus, 48K RAM, and Disk drive. With DOS and the LISP nucleus in core, 24K of memory remains for user programs. The LISP nucleus is represented by 73 commands.

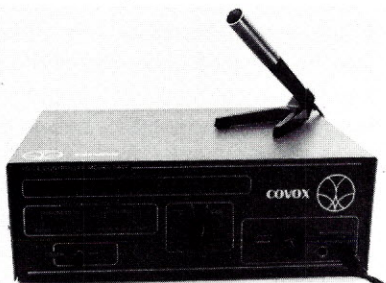
Supplied with the system are a series of utility packs. With the utilities in core, 26 extended LISP commands also become available.

App-L-LISP also provides an editor, which includes Global Replace and a Pretty Printer. Included is a utility that enables your program to be Pretty Printed out to a text file, for editing by a generalized text editor. The system uses a hashing scheme upon file load, so data is not repeated in memory.

For more information, contact: Datasoft, Inc., 19519 Business Center Dr., Northridge, CA 91324, phone: 213/701-5161. **CIRCLE 9**

Speech Analysis is Based on Human Speech Processing Model

A new, self-contained speech analyzer and voice controller is offered by Covox Company. Rather than analyze the speech waveform, the Covox Model I Voice Controller extracts speech clues in the manner



of a human listener. The Covox Model I identifies voice fundamental pitch and voicing duration, and it determines the voiced/unvoiced distinction in human speech.

Vowel cues are cross correlated with voicing to improve accuracy. Conventional radio and telephone channels suffice for this speaker independent system. Common kinds of noise clicks are suppressed, even when considerably more intense than the message signal. Priced about \$400, 16 separate and distinct commands are recognized with unlimited expansion capability when used with a micorcomputer. The Model I is also suitable for battery power. For further details and listing of topics in the research monograph *The Bionic Ear*, contact Covox Co., P. O. Box 2342, Santa Maria, CA 93455, phone: 805/937-9545 or 928-4818. **CIRCLE 10**

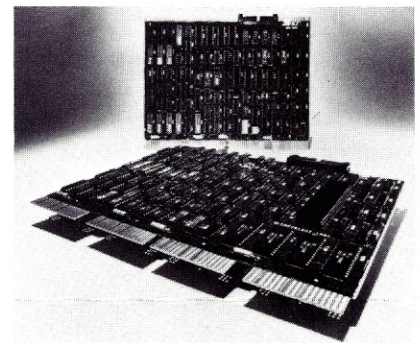
Array Processor for LSI-II

SKY Computers, Inc., announces

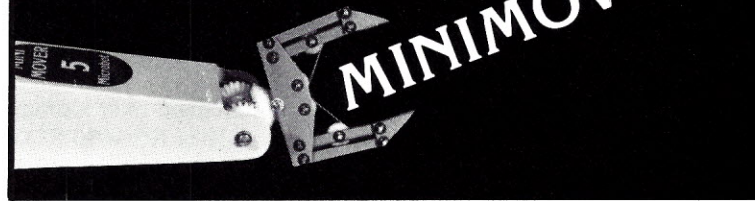
production of the SKYMNK "Micro Number Kruncher" Array Processor. The SKYMNK is a fully programmable floating point array processor designed specifically for microcomputers.

It is designed to provide high-speed (up to one megaflop), floating point processing for applications ranging from radar, sonar, seismology and tomography to speech recognition, process control, and robotics.

The SKYMNK is an entire floating point processor on two quad PCB boards that plug into any LSI-11 or 11/23 quad Q-bus backplane. It



For Robotics Research, Education, ... and Pure Enjoyment



THE LOW-COST MINIMOVER-5 TABLETOP ROBOTIC ARM BY MICROBOT — \$1695.

USE IT AT —

- The Lab
- Classroom
- Commercial Environment—or simulate Industrial Robots
- Workshop—with your Personal Computer

BUY IT —

For the cost of a good peripheral

*Apple II is a trademark of Apple Computers, Inc.

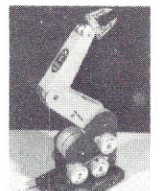
INTERFACES WITH —

Apple II*, PMC-80, TRS-80 (Model I/Level II) computers

MICROBOT DISTRIBUTOR



DORING ASSOCIATES
1744 RTE. 9
CLIFTON PARK, N.Y. 12065
(518) 371-9499



CIRCLE 25

operates under RT-11 or RSX-11M for FORTRAN or Macro programs, is capable of computing vector math, Fast Fourier Transforms, digital filtering, format conversions, and image processing at speeds 50 to 100 times faster than the stand-alone microcomputer.

The SKYMNK operates internally with standard PDP-11 32 bit single precision floating point format and 48 bit extended precision operation. It is tightly coupled to the host computer and shares the host's memory (up to 1 megabyte addressable).

It also extends the LSI-11's instruction set to include vector, matrix and compound mathematical instructions computed in real and complex arithmetic.

SKY Computers has also developed a software simulator, written in FORTRAN, which provides for off-line software development and a soft-fail mechanism for the hardware.

The SKYMNK has a list price of \$5990 and is available in OEM quantities for less than \$4K. Its delivery time is 90 days. The simulator is available for the cost of the media, handling and shipping (\$25 prepaid).

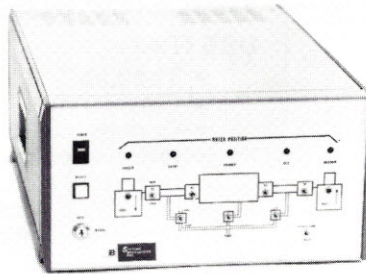
For more information, contact: Howard Klemmer, SKY Computers, Inc., P. O. Box 8008, Lowell, MA 01852, phone: 617/454-6200.

CIRCLE 11

Programmable Control Unit

Systems Innovations, Inc. announces the VI μ P Series 9000 programmable control unit. The 9000 was developed for semi-custom applications in machine automation, process and industrial control, instrumentation and robotics. At the heart of the system is the CPU 65/08

6502 based microcomputer with up to 24K of ROM/EPROM and 8K of RAM. Optically isolated I/O, analog I/O, stepping motor drivers and custom control boards can be



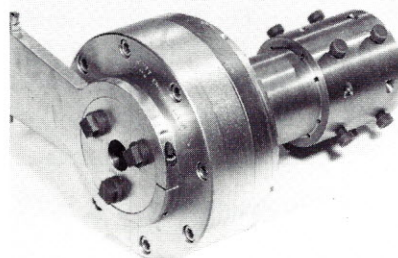
included in the system. Power supplies for logic and analog circuits are available in addition to 1 to 7 amps of bulk power for motors, relays and other user machine functions. Programming can be done in BASIC, FORTH, PL65 or Machine Language. A large uncommitted area allows auxilliary circuitry to be built in. User features such as front panel control and graphics, keyboards, and alphanumeric displays can be specified as part of a full custom system.

For further information, contact: Norman R. Prevett, Systems Innovations, Inc., P. O. Box 2066, Lowell, MA 01851, phone: 617/459-4449.

CIRCLE 12

Rotary Timing Valves

Rotary timing valves from Scott Engineering combine the functions



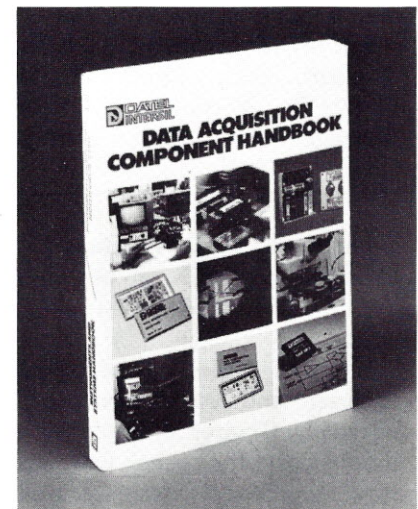
of a rotary union with those of a timing device, such as cam-operated micro-switches. Suitable for low or high pressure liquids or gases, these devices provide ON/OFF switching as equipment rotates.

The valves are mounted on the end of the rotating shaft. They can be designed with an extension, as shown in the photograph, to place the hose connections inside the bearings of the rotating element. This arrangement is designed for disassembly for inspection and maintenance.

For further information, contact: Scott Engineering Co., 430 N. 9th St., Olean, NY 14760, phone: 716/372-5690.

CIRCLE 13

Free Literature



A new Data Acquisition Components Handbook is offered free by Datel-Intersil. The 566 page handbook contains information on the following monolithic, hybrid and modular products: A/D and D/A converters, data acquisition systems, sample-holds, operational amplifiers, instrumentation amplifiers, multiplexers, special

BOOKS

Robotics Magazines from Britain

Two new robotics-related magazines are available from Britain. IFS Publications, which publishes the *Industrial Robot* and organizes seminars and conferences in the UK, has introduced *Assembly Automation* and *Sensor Review*.

In articles, interviews, and news, *Assembly Automation* deals with modern assembly methods and machinery. News briefs and short articles are the magazine's mainstay. Most are written expressly for the magazine, though reprints do appear. Taking the February 1981 issue as an example, I found five technical features from various corners of the world: one apiece from Italy, England, and the Soviet Union, and two from Japan.

The magazine gives a calendar of events, mostly on European conferences, although they do cover some of the important US and

Japanese gatherings. The February issue contained a good preview of AUTOMAN '81, held May 19-21 in Brighton, UK.

Another section of *Assembly Automation*, called "Inside Japan," is written by Associate Editor, J. Hartley, based in Tokyo. Hartley reports on the Japanese approach to automation, enriched with specific examples of the work being done there.

Assembly Automation has a "New Products" department, as well as a section giving the abstracts of recent technical papers. They also review conferences. My sample issue reported on the highlights and some of the papers given at last November's "Autofact West."

Sensor Review is similar in style to *Assembly Automation*, but focuses on intelligent sensory systems. Among other topics, the January 1981 issue contained articles on speech recognition, multiple sensors

for a mobile robot, and on using voice and laser sensors for semi-automatic sorting. Like the articles in its sister publication, these are written mostly by Europeans, though some US and Japanese technology is covered. Like *Assembly Automation*, *Sensor Review* also gives industry news, events, and new products.

Assembly Automation and *Sensor Review* are welcome new technical publications. Though one can obtain most of their material through sources in the US, I found it refreshing to get a British and European perspective on robotics. Both magazines are well written and have good graphic presentation. They both are published quarterly and cost, outside of England, £42 (about \$84 US dollars) for one year's subscription.

—Reviewed by Jerry Saveriano

function devices, and power supplies.

All products are organized into selection tables and are categorized by function and performance. Data sheets are included in the catalog for key products. Ordering information is also included.

For more information, contact: Ted Petit, Systems Applications Engineer, Dattel-Intersil, Mansfield, MA 02048, phone: 617/339-9341, ext. 221.

CIRCLE 14

Publications Guide

Westlake Subscription Service has released its Spring WESTLAKE GUIDE, descriptions of more than 350 computer, technology,

electronics & business magazines, newsletters, and report/database services. The new 24-page catalog features the Westlake Trial Plan in which interested readers may sample listed publications for a handling fee refundable if they decide to subscribe. For a free copy of the Guide, send your name and address to Westlake Subscription Service, 4200 So. Louise Ave., Ste. 104, Sioux Falls, SD 57106, or call 605/331-6930.

CIRCLE 15

Robot Institute Issues Two Position Papers

With competition between the U.S., Germany and Japan in mind, the Robot Institute of America (RIA)

has issued position papers on new robot advancements and improvements for U.S. industries. Currently, two papers, *The Decline of Productivity and the Resultant Loss of U. S. World Economic and Political Leadership* and *Robotics Technology—A Major Component in the Solution of the U.S. Productivity Problem* are available. These papers concern themselves with the practical and economic justification of how industries can improve productivity in the United States.

To receive copies of these position papers, write to: RIA Headquarters, One SME Dr., P. O. Box 930, Dearborn, MI 48128, attention: Position Papers, 313/271-1500, ext. 410.

CIRCLE 16

MEDIA SENSORS

Business Week, June 1, 1981, **Better Productivity is the New Priority.** The highest levels of management have given productivity top priority. Last year, for example, Westinghouse established a 250-person Productivity Center and committed itself to increasing both factory and office productivity by 50%. To meet this goal in the factory, Westinghouse is depending on robots. Last year, they installed 50 robots, and this year they plan to add 200 more.

The robotization plans at General Motors and General Electric outstrip even those of Westinghouse. By 1983, GM plans to add 2,000 robots to the 300 machines already in operation. By 1990, they expect to have 14,000 steel-collar workers. Meanwhile, GE plans to transfer the tasks now done by 37,000 blue-collar workers to 15,000 robots. With this kind of investment in automation, the number of U.S. robots sold each year could rise from the current 2,000 to 100,000 by 1990.

Added to the explosion in robotics is the spreading use of computer-aided design (CAD) and computer-aided manufacturing (CAM) systems. Sales of CAD systems are expected to grow 40% annually to \$3 billion by 1985. And by 1990, CAM sales may be even greater than CAD. Most importantly, CAD and CAM systems are beginning to "talk" to each other—leading to computer control of the whole manufacturing process. We are approaching the time when, according to William Beeby of Boeing, "CAD and CAM are going to revolutionize the way we do business."

Ron S. Heinzl, *Los Angeles Times*, June 7, 1981, **Here's Your (Beep!)**

Burger and Coke. Have you ever patiently waited for your food at a restaurant—while it seemed that your waiter was acting like a robot? Soon there will be waiters that act like robots because they are robots.

The Burgerworld International chain plans to staff a prototype restaurant near Windsor, Ontario, with three \$20,000 serving robots. Customers will order food by intercom directly from the cook. When their food is ready, the cook programs a robot to serve it. The robot waiter, looking "something like R2D2," can carry four trays at once and serve nine tables in 72 seconds.

Los Angeles Times, May 31, 1981, **Talking Elevator Opens Door to Future.** At the headquarters of Westinghouse, elevators talk to their passengers. Among other messages, an elevator says, "Please let the door close," if a passenger holds it open too long. If someone enters an elevator and forgets to push a button for a floor, it says, "Please select your floor," and after the passenger does, it says, "Thank you."

Westinghouse hopes its MicroPhonic-60 voice synthesis system will aid blind or forgetful people. But one of the system's creators, Alan Mandel, sees it more as a precursor of widespread microprocessor-based speech technology. As Mandel puts it, "One day a voice from the dashboard of your car is going to tell you when your oil is low or when your alternator is going bad. Your oven is going to tell you when to baste the turkey, and how to make your grandmother's favorite recipe."

There have been talking elevators—using tape recordings—for years. But electronic speech synthesis has the advantage of

needing no moving parts. More importantly, computer-based speech synthesizers may someday be able to construct sentences after being "taught" a vocabulary. With that kind of flexibility, machines—especially computers—will begin to communicate with people in human language. As Mandel comments, "People will see ever-more-friendly computers."

Mother Jones, May 1981, **Steel Worker Blues.** After decades of threatening, robots may finally be invading the U.S. labor market in the 1980s. These computer-controlled co-workers will soon be taking over automobile welding, "hazardous work like spray painting," and finally, assembly line work.

These robots, which the article calls "mutant fire-hydrant look-alikes," range from \$10,000 to \$80,000—and may even be rented, from a Japanese consortium.

For the reader worried about losing his job to a robot, *Mother Jones* has this advice: "there's always the chance that you'll find a job somewhere as a floppy disc-washer."

Pamela Weintraub, *Discover*, June 1981, **Raising the Robot's I.Q..** On the barren, jagged moonscape, someday, robots will be mining ore, transporting it to smelters, and assembling the smelted metal into spaceships, satellites, and—new robots. NASA is seriously discussing robot-run factories on the moon. But whether we speak of NASA's robots or the "one-armed welders and bolt-tighteners" in our factories, one thing is clear: we need more intelligent robots, those that can—to some extent—see, hear, feel, smell, communicate, and carry out high-

level decisions.

Developing vision systems is an important step towards this goal. Presently, even those robots in industry that can see, see only a two-dimensional world. In laboratories, such as the National Bureau of Standards (NBS), researchers work on getting robots to see in three dimensions. At NBS, a plane of light shines from the robot's fingertips, reflects off nearby objects, and is sensed by a wrist-mounted camera. Using geometry, a computer then calculates the shape and location of the object and tells the robot how to grasp it.

But robots using light beams are limited—they can only see for a few feet and only in the direction the light shines. Ideally, a robot would see stereoscopically, with two eyes, as humans do. Unfortunately, present-day robot eyes can't duplicate the complexity of human vision. After all, a human has 150 million light sensors in the retina, sending signals to billions of nerve cells in the brain.

Researchers try to imitate human vision in robots, but without the full complexity of the human vision system. Tom Binford of Stanford University, for instance, uses two cameras to see in stereo, then, by computer, reduces the sensed image to lines that represent the most important edges and curves in the image. To recognize images, Binford's robot is building a world model that consists of abstract representations of common objects. Since the robot must process millions of bytes of data to reduce the image to its features and then to compare these features with stored models of objects, it takes as long as two to three minutes to recognize even simple objects. Only when future computers can process data

1000 times more quickly will robot eyes begin to compete with their human counterparts.

Even with advanced vision, robots will still need other senses. Danny Hills and John Hollerbach at MIT are giving robots the sense of touch through "artificial skin." The skin consists of thin layers of rubber, each lined with wires. When the skin touches an object, the layers press together, allowing a greater current to flow through the wires. The greater the pressure on the skin, the more current flows. The MIT touch sensor forms an image—in many ways like a visual image—of the object it touches.

In addition to touch, a robot needs to measure the forces exerted on its wrist. One of the most sophisticated force sensors has been installed at the Charles Draper Laboratory. In this system, three lights shine on three light detectors that are mounted on the robot's hand. The hand, in turn, connects to the robot's wrist through a joint made of three springy cylinders. As the hand exerts force on an object, the cylinders contract, stretch, or twist—causing the lights to move across the faces of the light detectors. A computer uses the lights' movement to calculate the force on the robot's wrist.

Raj Reddy, director of Carnegie-Mellon's Robotics Institute, predicts that future robots will have a whole spectrum of sensors. Their ears will listen for breaking parts and respond to human commands. Their chemical sensors will have certain powers of taste and smell. Their infra-red or ultra-violet sensors will see in the dark. And their ultrasound or sonar sensors will help them navigate deep in the ocean.

Some of these sensors are being used today by Hans Moravec in the

mobile robot he is building at Carnegie-Mellon. Moravec's previous robot, built at Stanford, could navigate across the room, using vision as its guide. The Carnegie-Mellon robot, though, should travel ten times as fast as the Stanford one. The difference is a powerful computer dedicated to quickly analyzing visual information.

Yet even with sophisticated sensors, robots still need limbs as versatile as human limbs. Ken Salisbury at Stanford, for example, is trying to improve the robot's hand. The one he is now building has three fingers, each of which can oppose the other two and grasp objects as the thumb and forefinger do.

Stanford's Tom Binford is also improving the robot's dexterity—coordinating two hands to perform a single task. Meanwhile, Robert McGhee at Ohio State University has built a six-legged walking robot. And Haruhiko Asada at Carnegie-Mellon is building a fast robot arm driven by ultra-light motors. By using materials such as titanium, fiberglass, and plastic, he plans to bring the arm's weight down to just thirteen pounds.

To increase robot intelligence, researchers are depending on advances in very large scale integrated (VLSI) circuit design. VLSI chips promise to improve processing speeds and information storage capacities by factors of a thousand over current microprocessors. By the end of this decade, roboticists believe that VLSI will provide every sensor on a robot with its own powerful, dedicated microprocessor.

To fully use the power of VLSI, roboticists are developing advanced techniques to give robots the power

(continued page 53)

ORGANIZATIONS

SME Forms New Association

SME has announced the establishment of the North American Manufacturing Research Institution of SME, or NAMRI/SME, an association of individual members engaged in manufacturing research and technology development. The SME Board of Directors approved the formation of NAMRI/SME at its April 27 annual meeting.

The primary objectives of NAMRI/SME are to promote and stimulate research, writing, publishing and dissemination of new manufacturing technology, to sponsor an annual Research Conference, and to coordinate efforts and cooperate with counterpart organizations worldwide.

Dr. Jiri Tlustý, and SME member and the Chairman of the NAMRI/SME. Dr. Tlustý is Professor of Mechanical Engineering and Chairman of the Metalworking Research and Design Group at McMaster University, Hamilton, Ontario.

Prab Changes Corporate Name

The corporate name of Prab Conveyors has been changed to Prab Robots. The name change officially took effect on June 1, 1981, after the company's shareholders approved the move at their annual meeting on May 29, 1981. The firm's conveyor and chip processing business will continue to be operated by Prab Conveyors, Inc., a wholly-owned subsidiary.

Prab Conveyors, Inc., was founded in 1950 by two men whose combined initials established the name "Prab." They added

"Conveyors" to denote their major business activity.

Prab's current management bought the company in 1961 and moved it from Detroit to Kalamazoo, Mich., Prab's present corporate headquarters. Before the end of that decade, the company expanded its basic conveyor lines to include metal scrap processing systems and, in 1968, an industrial robot. This unit was designed specifically to fill applications in the die casting industry, which was a major conveyor and scrap processing systems customer for Prab.

In 1979, Prab acquired the Versatran line of industrial robots from AMF, Incorporated. Now called the Prab Models E, FA, FB and FC, the Versatran Robots were applied to tasks such as machine loading and unloading, investment casting, spot welding, forging, stacking and de-stacking, and palletizing. Now with the explosive growth in the robot industry, Prab has changed its name to identify itself with robotics.

Calendar

The **Robots VI** Conference and Exposition, at Detroit's Cobo Hall, March 1-4, 1982, may have exhibits from every major U.S. robot manufacturer and from several European and Japanese firms as well.

The concurrent conference program will cover topics such as assembly, welding, die casting and foundry operations, aerospace applications, machine loading and unloading, material handling, vision

and sensors, and research and development. In addition, there will be sessions focusing on the human factors associated with robotics including worker and management attitudes and safety in the workplace.

Complete conference details will be announced later this summer. The Robots VI Conference Chairman is George Munson, Vice President of Systems Division, Unimation, Inc., Danbury, Conn.

For further details about Robots VI, contact RI/SME, One SME Dr., PO Box 930, Dearborn, MI 48128; phone 313/271-1500, ext. 416.

The **AUTOFACT III** Conference and Exposition at Cobo Hall, Detroit, November 9-12, 1981, will offer the most comprehensive look yet at both present automation technology and the near future when highly automated, computer integrated factories may be commonplace.

Sponsored by the Computer & Automated Systems Association of the Society of Manufacturing Engineers (CASA/SME), the 3½ day convention is expected to attract more than 6,000 manufacturing executives and engineers to the sessions and equipment exhibits.

The exhibits will include CAD/CAM equipment—such as control systems, graphics, and software programs—automated assembly and fastening, computerized storage/retrieval and material handling, industrial robot systems, diagnostic monitoring, computer-based testing and measuring, and on-line inspection systems.

To receive complete details on the conference, exhibits, registration and hotel information, write to

AUTOFACT III, SME Public Relations, One SME Dr., PO Box 930, Dearborn, MI 48128.

The University of Maryland offers an **Industrial Robotics** course, August 3-7, 1981. It will be held at University College's Center of Adult Education, College Park, Maryland. Among other objectives, the course should give participants a comprehensive overview of current and projected industrial robot applications, and to provide a state-of-the-art review of advances in robot programming, control systems and theory, brain modeling, artificial intelligence, robot vision, and CAD/CAM systems. Among the

eight robotics experts lecturing at the course are featured lecturers Dr. James Albus, Dr. Carl Sagan, and Mr. William Tanner. For more information, contact: The Conferences and Institutes Program, 301/454-4712.

Previously Announced:
Refer to the original Robotics Age Calendar announcement for details.

IEEE Computer Society Conf. on Pattern Recognition and Image Processing, Aug. 3-5, 1981, Dallas, TX. Announced in Vol. 2, No. 3.

7th International Joint Conference on Artificial Intelligence, Aug. 24-28, 1981, Vancouver, Canada. Announced in Vol. 2, No. 3.

Media Sensors

(continued from page 51)

to make plans and adapt and react to the environment. Groups like the one at Carnegie-Mellon are building abstract, problem-solving models that may give robots some of the reasoning power of humans.

In five years, Raj Reddy asserts, a space robot may be able to navigate its way from the space shuttle to a satellite. In ten to fifteen years, robots will be autonomous enough to navigate over rugged terrain and to assemble complex structures. As James Albus of NBS comments, "There are two illusions about robots. One is that they are already here. The other is that they'll never be here..."

BACK ISSUES OF ROBOTICS AGE

Did You Miss Any?

SUMMER 1979: Digital Speed Control of DC Motors; Industrial Robotics '79: Introduction to Robot Vision; The Grievet Chess Playing Arm; Robotics in the Soviet Union; The Robotics Age Competitive Event.

WINTER 1979: Advances in Switched-Mode Power Conversion (Part I); Prospects for Robots in Space; Robotics Research in Japan; Report from IJCAI6.

SPRING 1980: Microcomputer Based Path Control; Robotics Research in France; Multiple Sensors for a Low-Cost Robot; The Robots of Autofact II; Inside Big Trak.

SUMMER 1980: Industrial Robots: Today and Tomorrow; Introducing the MiniMover 5; Advances in Switched Mode Power

FALL 1980: Using Optical Shaft Encoders; Interview with Victor Scheinman; Robot Vision for Industry; The "Autovision"™ System; Industrial Robotics '80; Robots on Your Own Time; Superkim Meets ET-2.

JAN/FEB 1981: A Robot Arm Without a Budget; An Interview with Joseph Engelberger; Robots V—Dearborn 1980; Robots on Your Own Time; Opto "Whiskers"; Robot Toy Designs.

MAR/APR 1981: Video Signal Input; Chain-Code; Camera Geometry for Robot Vision; TIG Welding with Robots; Robot Digestive Tract; Robots On Your Own Time.

MAY/JUNE 1981: Rehabilitative Robots; A Homebuilt Computer Controlled Lathe; An Interview with Charlie Rosen; Superkim Meets ET-2, Part II.

Send in your name, address, and *payment* to receive any back issues of ROBOTICS AGE you've missed. Payment must accompany order—back issues cannot be billed.

I would like to receive

_____ Copies of SUMMER 1979	
_____ Signed, numbered Collector's Edition	\$20.00/ea.
_____ Xerox copy of original	\$ 7.50/ea.
_____ Copies of WINTER 1979	\$2.50/ea.
_____ Copies of SPRING 1980	\$2.50/ea.
_____ Copies of SUMMER 1980	\$2.50/ea.
_____ Copies of FALL 1980	\$2.50/ea.
_____ Copies of JAN/FEB 1981	\$3.00/ea.
_____ Copies of MAR/APR 1981	\$3.00/ea.
_____ Copies of MAY/JUNE 1981	\$3.00/ea.

Total postage/handling _____

Total payment enclosed _____

ROBOTICS AGE

Post Office Box 725 La Canada, California 91011

Add \$1.25 postage and handling for the first back issue, each additional back issue add 50¢.

68000+PASCAL+^{TeleSoft} ADA⁵= POWER

THE IN/7000D PUTS ARTIFICIAL INTELLIGENCE WITHIN REACH...

68000 BASED IN/7000D

Drastically increased processing speed and flexibility:

- 8 MHz 68000 CPU
- 32-Bit internal arithmetic registers
- 24-Bit address register
- Powerful assembly language instructions support modular programming
- Designed for expansion to 32 bit word size
- 8 Levels of interrupt priority
- Vectored interrupts and DMA fully supported
- Outbenchmarks the IBM 370/145²
- High speed string processing

Dramatically increased memory:

- 68000 directly addresses 16 MB of memory
- Sorts can be done in core rather than in disk I/O

Compatibility:

- Pascal to 68000 Native-Code Compiler
- Uses DEC Q-BUS³ and standard DEC³ compatible peripherals
- Ability to mix a variety of CPU's in the same system. Currently supports any DEC LSI-11³ or MOTOROLA 68000
- Forthcoming is Intel Multibus⁴ Version

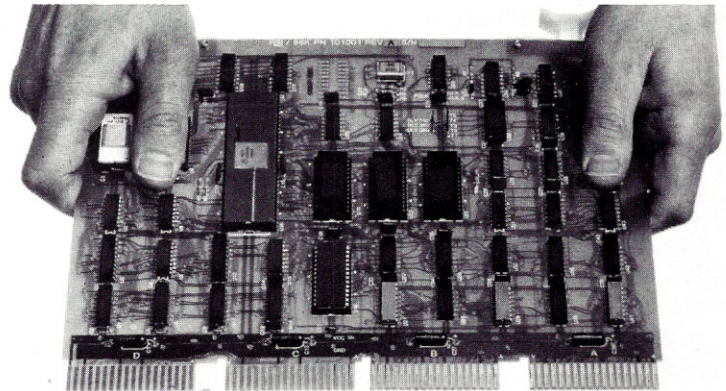
PASCAL

- Fast program development
- Self documenting
- Supports structured programming
- Easy to update
- Easy to maintain
- Transportable from computer to computer
- Powerful logical constructs greatly simplify programming

For example: Modular procedures and functions

Strong data types
If-then-else structures
Case structures
Do-while structures
Repeat-until structures

- Local and global variables
- Recursive problem solving
- Block insert—a block of statements may be inserted anywhere one statement can exist
- Built in Boolean operations: end of file, end of line
- Library capability
- Program segmentability
- Procedure linking
- RS1¹ Pascal compiles to native 68000 code
- Built-in powerful string-handling features



R&D ENGINEERS — — — GET YOUR HANDS ON IT!!!

QUANTUM LEAP IN PERFORMANCE OVER THE 8-BIT AND OLDER 16-BIT PROCESSORS!

Gives you overall speedup of robotic feedback cycle time.



Desktop version available now!

Saves development time.

- High speed sensory data processing
- High speed string processing power
- Fast coordinate transformations
- Easy implementation of in-memory AI algorithms, predicate calculus and trajectory computations
- Design and test algorithms quicker and easier
- Both Pascal and 68000 support features that make debugging far more efficient
- Plenty of memory, no need to use extra time for "programming tricks," previously needed with limited memory
- Mixed mode listing (Pascal source statements followed by 68000 statements)

¹"Kilobaud Microputing" October, 1980

²A trademark of Renaissance Systems, Inc.

³A trademark of International Business Machines

⁴A trademark of Digital Equipment Corporation

⁵A trademark of Intel Corporation

⁶A trademark of Department of Defense

Dealerships
are available.

Call us now for more information, or fill in the coupon. (301)840-0700

I AM INTERESTED! Send more information now.

Name _____ Title _____

Company _____

Street _____

City _____ State _____ Zip _____

Phone _____ Call me IMMEDIATELY ☐

My need is: ☐ 1-3 mo. ☐ 3-6 mo. ☐ Future project ☐ File

☐ I am interested in a dealership.

INTELLIMAC/Intelligent Machines

51 Monroe St., 18th Floor • Rockville, Md. 20850
Ph. (301)840-0700 • TWX: 710-828-9786 • Cable: EAIL ROCKVILLE MD

INTELLIMAC 
Intelligent Machines

51 Monroe St., 18th Floor
Rockville, Md. 20850
(301)840-0700
TWX: 710-828-9786
CABLE: EAIL ROCKVILLE MD